Aalto University
School of Science
Degree Programme in Computer, Communication and Information Sciences

Fatiqa Nadeem

Impact of Small-scale Agile Development Projects on Customer Satisfaction in a Software as a Service organization

Master's Thesis

Espoo, 20th September, 2020

Supervisor: Professor Marjo Kauppinen, Aalto University
Instructors: M.Sc. (Tech.) Ville Kivi, Aalto University

AALTO UNIVERSITY
School of Science
Master's Programme in Software and Service Engineering

ABSTRACT OF THE
MASTER´S THESIS

Author:
Fatiqa Nadeem

Title:
Impact of Small-scale Agile Development Projects on Customer Satisfaction in a Software as a Service Organization

Number of pages:	Date:
102	22-09-2020

Major or Minor:
Computer, Communications and Information Sciences

Supervisor:
Professor Marjo Kauppinen

Thesis advisor:
Ville Kivi, M.Sc. (Tech.)

Software companies are nowadays adopting Software as a Service (SaaS) delivery model. In the SaaS delivery model, two important sources of revenue are retaining existing customers, aka customer retention, and gaining more customers via a positive 'word of mouth'. Customer satisfaction plays a vital role in customer retention. Customer satisfaction can be achieved by providing a service that creates value for the customer and this eventually leads the company to establish a loyal relationship with the customer. To achieve customer satisfaction, SaaS companies are continuously striving to implement practices that will help them in delivering quality services to their customers.

This thesis addresses the research problem *"How can SaaS companies execute small-scale agile development projects in order to maximize customer satisfaction?"* To answer the research problem we used an action research method for a Finnish company operating on the SaaS delivery model. The data collected as a part of the empirical study was later analyzed using the Grounded Theory. In our literature review, we focused on understanding the key concepts such as the SaaS delivery model, customer satisfaction, common challenges that agile development projects face, and good practices that can be used to eliminate challenges. The focus of the empirical study was to understand the current state of small-scale agile development projects at the case company, the impact these projects have on customer satisfaction, and the challenges that developers face when working on such projects. Five key challenges uncovered during the research were delayed project delivery, poor communication between the development team, and an undefined process for project delivery. Based on the literature review and the results of the empirical study this thesis suggests good practices that development teams can use in order to achieve maximum customer satisfaction. A few of the good practices suggested by this thesis are such as creating user stories, implementing pair programming, and providing guidelines to help developers steer their projects towards completion.

In conclusion, developers should work, along with the customer, on eliciting requirements towards the beginning of the project by creating user stories and gain a deeper understanding of the customer requirements. As per existing literature collaboration with the customer to create user stories will eventually result in fewer post-development changes requested by the customer and hence maximizing customer satisfaction.

Keywords:	Publishing Language: English
Software as a Service; customer satisfaction; small-scale agile development; user stories; pair programming; customer success; customer loyalty; customer retention.	

Acknowledgements

You read to become all knowledgeable
Stop seeking all this knowledge, my friend
Only an Alif is what you need !

پڑھ پڑھ علم تے فاظل ہویا
علموں بس کریں او یار
اکو الف ترے درکار

———

Aik Alif by Bulleh Shah

I had never even considered enrolling in a master's degree let alone moving 4695 kilometers away from my home. The reason I was able to achieve the unknowing is the faith my parents showed in me even when they were diametrically opposed to my decisions. Also because my fiance, Saddam, is a keen listener and supports my erratic decisions with as much zest as my parents.

During my bachelors I had the pleasure of working with Asst. Professor Majid Maqbool and he made me understand that software is a lot more than just a few lines of code. Crawling out of university I got a job at CureMD, Pakistan and got the opportunity to work along with some of the finest human beings. One of them was Mr. Usman Riaz, a person who always motivated me to achieve my potential. In my four years as a professional I have yet to find a better team lead than Usman. Also, thank you Mr. Adeel Rafiq for believing in my almost non-existent leadership abilities. The workshops with you helped me shape my personality and unknowingly prepared me for a new life somewhere swamped in snow.

My professional experience motivated me to learn more about software engineering and was the key stimulus behind applying at Aalto University. Two years at Aalto University helped me in growing on a professional as well as a personal level. This thesis was a valuable experience for me and a great opportunity to understand the concepts of customer satisfaction & agile development.

I thank my supervisor, Professor Marjo Kauppinen, for all the guidance and help she provided during this thesis. Thank you Marjo for being extremely supportive and kind during the global pandemic, Covid-19. I consider myself lucky to be a student of Professor Marjo Kauppinen and Asst. Professor Majid Maqbool.

Finally, I want to thank my friends for providing their support during all my endeavours. Thank you Rabia for being my shrink and my healer. Thank you Maria for motivating me to apply at Aalto University and helping me every step of the way. Thank you Danish and Nameer for making me laugh at work during dark times. And lastly thank you Ahsan for helping me write the punjabi verse at the beginning of this page.

Espoo, September 22, 2020
Fatiqa Nadeem

Table of content

List of Images

List of Tables

1 Introduction

This chapter explains the motivation for the Master's thesis. The research problem, objectives, and structure of the thesis are also described in this chapter.

1.1 Motivation for the research

Software as a Service indicates a model where the software runs on a cloud and multiple users can access a single instance of the application via an internet connection *(Aleem, et al., 2018)*. Software providers are widely adapting *Software as a Service (SaaS)* delivery model due to its widespread benefits for both the provider and the consumer. One of the benefits being the ease to deliver services to the consumers all across the globe *(Aleem, et al., 2018)*. This ease of reaching globally, however, also adds complexity to the overall *SaaS* design as such products need to cater to the requirements of *multiple* stakeholders *(Aleem, et al., 2018)*.

In comparison to traditional web-based applications, the SaaS delivery model *also* focuses on the monitoring of the product after the initial implementation for the customer has been completed. One important source of revenue for SaaS companies is retaining their customers and gaining new customers based on the 'word of mouth' from their existing customers *(Mehta et. al, 2016)*. Hence customer satisfaction is of utmost importance for SaaS companies as it has a direct impact on the company's revenue. As per *Oliver (2010)*, delivering a product or service that meets the customer's needs eventually results in customer satisfaction. In this modern era customers are looking for products or services that add value to their lives *(Grönroos, 2007)*. Hence, *quality* is of utmost importance in a SaaS product and should be incorporated in the product from the *initial* service design phases such as code architecture design and user-interface design (*H. Kashfi, 2017)*.

SaaS companies nowadays strive to understand and implement processes that will help them in delivering quality products or services to their customers. Hence, design plays an important role towards the overall success of a SaaS product *(Vidhyalakshmi and Kumar, 2014)*. However, SaaS companies face multiple challenges while trying to design and develop a product that meets their customer needs. One of these challenges is working with a globally distributed team *(Fitriani et. al, 2016)*. Most software companies are working with a geographically distributed team nowadays. In practical terms this implies that the team developing the software is located in different geographic locations or timezones *(Ibrahim et. al, 2016)*. This global distribution of teams makes team management a challenging task *(Fitriani et. al, 2016)*. Hence the motivation behind this thesis is to highlight the challenges that software developers face while developing a SaaS product and the impact of these challenges on customer satisfaction. Furthermore, this thesis also highlights good practices that software designers and developers can incorporate during the development process in order to maximize customer satisfaction.

1.2 Research problem and questions

The main aim of the thesis is to define a set of good practices that should be followed during small-scale agile development projects within a SaaS company. The research problem can be defined as follows:

How can SaaS companies execute small-scale agile development projects in order to maximize customer satisfaction?

An important perspective through which the main research problem is looked at is that of *customer satisfaction*. The research problem is further divided into two sub questions in order to evaluate the contrast between the ongoing small-scale development practices within the case company and the practices defined in the literature.

Research Question 1: *What is the current state of small-scale agile development at the case company?*

The first question is designed to evaluate the current state of small-scale agile development projects that cater to the changing needs of its stakeholders within the case company. Customer satisfaction is the main focus point when evaluating the current state. Under the current state we aim at understanding the current process used for small-scale agile development projects, the current challenges that these projects face and the impact of these challenges on customer satisfaction.

Research Question 2: *What good practices can be used by the case company in small-scale agile development projects to maximize customer satisfaction?*

The second question aims at contrasting the current practices with those defined in the literature in order to highlight good practices that will eventually result in maximum customer satisfaction.

1.3 Scope and objectives of the research

The main objective of this thesis is to analyze the current state of small-scale agile development projects at the case company, identify good practices from literature and map them in the current process so as to maximize customer satisfaction. The aim is to combine the academic research and good practices in order to define a set of practices that can help software designers and developers to maximize customer satisfaction.

The empirical study of this thesis focused on studying and evaluating the current state of small-scale agile development projects at the case company. Finally, the thesis compares the current state of the case company with the good practices defined in the literature. Through this comparison the main aim was to create a set of company-wide standard

practices in order to maximize customer satisfaction from small-scale agile development projects.

The results of this study can be utilized in all future small-scale agile development projects; and will help in creating a set of guidelines that software designers and developers can follow during such projects.

Table 1. Objectives of research questions

Research Question	**Objective**
RQ1: What is the current state of small-scale agile development at the case company?	Analyze the current process that is followed at the case company in small-scale agile projects. Furthermore, to analyze the role of software developers in such projects and the current challenges that are faced when it comes to customer satisfaction.
RQ2: What good practices can be used by the case company in small-scale agile development projects to maximize customer satisfaction?	Gather good practices that lead to customer satisfaction from the literature and incorporate them in the current process followed within the case company.

1.5 Structure of the thesis

The thesis begins with an introduction to the research domain, the motivation behind choosing the research domain, the definition and scope of the research problem. The second chapter of this thesis presents the research methodology used to conduct the literature reviews and the empirical research. The third chapter consists of the literature review that presents a deeper understanding of the key concepts related to the research problem. The fourth chapter details the structure of the case company and presents a detailed analysis of the data collected as a part of the empirical research. Moreover, the fourth chapter also enlists a set of good practices that can be used by the case company in order to eliminate the identified challenges. The fifth chapter is used to create a concise understanding of the main results of this thesis. It is mainly a discussion of the suggested good practices from the previous chapter and their impact on customer satisfaction. The sixth chapter concludes the thesis by giving concrete conclusions that can be used to improve the state of small-scale agile development and talks about future research areas that can emerge from this thesis.

First a thorough review of the existing literature is utilized to create a uniform understanding of the key concepts that were identified towards the beginning of this thesis. For the empirical part of this thesis we utilized *action research* as the main approach. Grounded theory is used to further categorize and evaluate the data collected from the empirical research. The understanding formed by reading the existing literature is then further utilized to support the conclusions made towards the end as well.

Table 2 summarizes the link literature review and empirical study share with each of the research questions.

Table 2. Structure of the thesis

Research Question	Literature Review (LR)	Empirical Study
RQ1	LR is used to create a deeper understanding of the key concepts of the thesis in order to facilitate the empirical research. Furthermore, LR is also used to understand the challenges that companies face in agile development projects.	During the empirical study we conducted interviews of a selected set of employees at the case company. This helped us in gaining insight of the current state of small-scale agile development projects at the case company.
RQ2	LR is used to identify good practices and support the reasoning behind the suggestions put forward by this thesis.	The results of the empirical study are used to suggest good practices that can eliminate the identified challenges.

2 Research Methods

This section describes how the research was carried out and the methods that were used for data collection and analysis.

2.1 Overview of the research approach and methods

The thesis aims at identifying the current challenges of small-scale agile development projects at the case company and propose practices that can be used to eliminate or reduce these challenges and maximize customer satisfaction. Keeping the purpose of this thesis in mind it was decided to choose *Action research* as the main research approach. Action research is cyclical in nature when it comes to linking literature with practical situations and aims at solving a problem based on present scientific research or relevant theories. *(Baskerville and Wood-Harper, 1996)*.

An important component of such projects are the humans that are involved throughout the project under various roles. Action research has grown its influence in the information systems domain during the 1990s *(Baskerville, 1999)*. This is because human interaction with information technologies cannot be ignored when studying information systems and action research aimed at understanding complex human processes *(Baskerville, 1999)*. According to *Hult and Lennung (1980)*, action research helps the researcher to understand the current situation and solve practical problems based on scientific knowledge.

According to *Baskerville (1999)*, the ideal situation to utilize action research needs to have the following three characteristics:

1. The goals of the researcher and the organization under study (known as the case company in this research) must be aligned. Also the research must be actively involved in the organization.

2. The results of the research should have the capability to be applied immediately in the organization
3. The research should link theory or literature with the practices with the organization.

The above mentioned characteristics are all present in the context of this thesis. Firstly, the researcher has been employed at the case company for more than a year and has participated in small-scale agile development projects under the role of *'Junior Technical Consultant'*. Secondly, the results of this thesis were presented in the form of concrete steps or actions that can be immediately applied in the case company. Thirdly, the overall research process used was iterative in nature and consistently links literature with practice. Figure 1 gives an overview of the three phases of action research used in this thesis.

Figure 1. Overview of action research phases used in this thesis.

Susman and Evered (1978) identified five cyclical phases of action research namely: diagnosing, action planning, action taking, evaluating and specifying learning. All these phases/activities are carried within the scope of a *client-system infrastructure (Susman and Evered, 1978)*. Client-system infrastructure here refers to an agreement between the researcher and the host organization that defines the boundary of the research *(Susman and Evered, 1978)*. Under the scope of this thesis we use a derivation of the cyclical phases specified by *Susman and Evered (1978)*. The three main phases used in this research were: *diagnosing, action planning* and *specifying learning*. Figure 2 gives an overview of the overall research process used to conduct this thesis and also maps each activity with the corresponding action research phases.

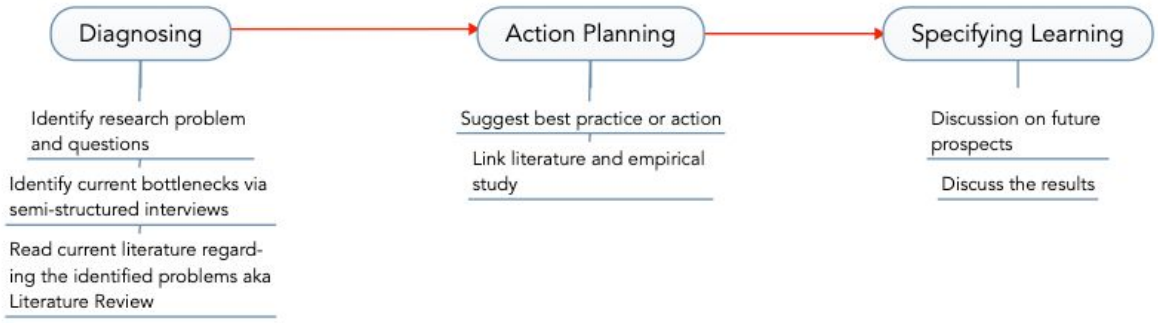

Figure 2. Overview of the research process

Diagnosing consisted of the identification of the primary problem or bottlenecks that were faced by the case company when it comes to small-scale agile development projects. *Action planning* phase corresponded with eliciting actions or practices that can be used to eliminate the problem. *Specifying learning* is an ongoing process that is used to discuss the findings of the research and elictate related future research prospects *(Baskerville, 1999)*. After the action planning was concluded, the identified challenges were discussed with the company supervisor and the concrete good practices that can be used to mitigate these challenges were also discussed. This discussion was done with an aim to specify the learnings of the thesis with the case company supervisor and gain their insight as well. The results of the thesis were also shared with company employees looking for future research prospects. Furthermore, the case company started a new research thesis based on one of the good practices suggested by this thesis.

Figure 3 provides a detailed summary of the research process. The figure has been color coded in order to make the mapping of the research process to action research phases more visible.

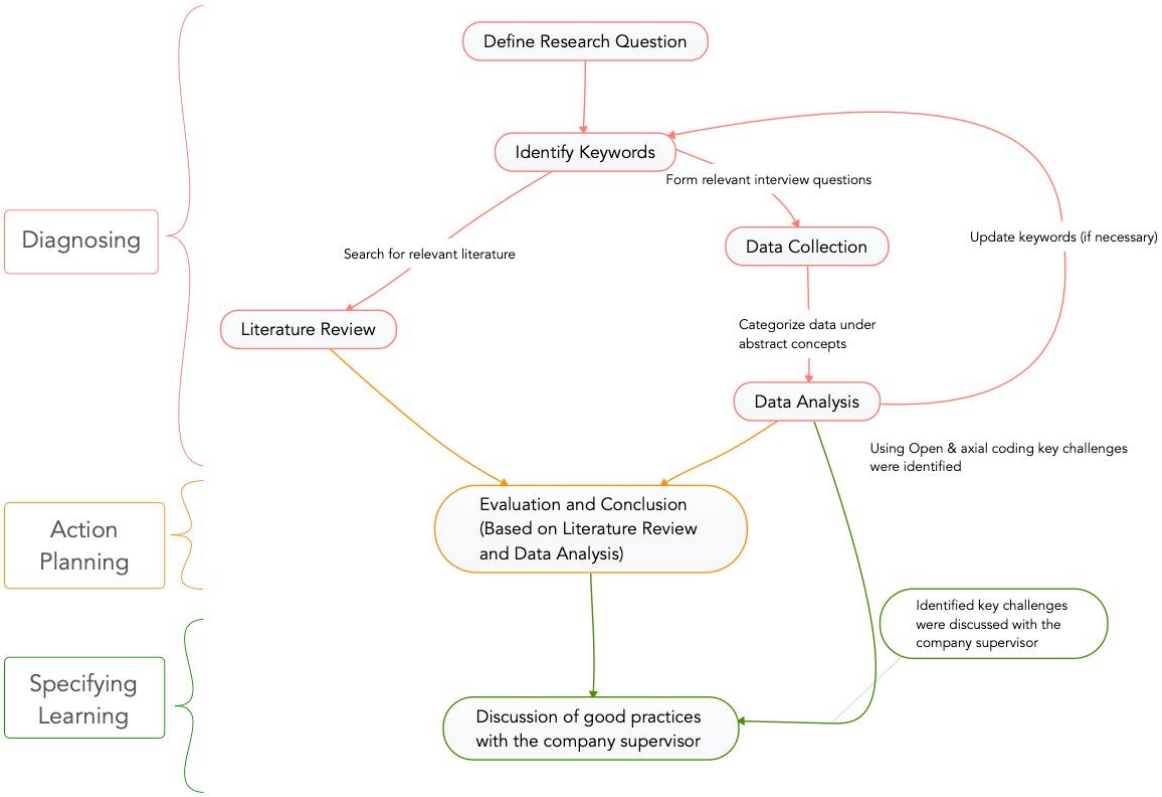

Figure 3. Detailed steps of the research process

At the beginning, the exact scope of the research problem was not clear. Hence, we decided to follow a *cyclical or iterative approach* and conduct the literature review after the empirical research. During the diagnosing phase, it was important to first understand the current state of the small-scale agile development projects at the case company. We started by identifying the research problem and questions by having a discussion with the company supervisor. Using the research problem and questions as the basis we identified the main keywords. *Individual semi-structured in-depth interviews* were chosen for data collection. The identified keywords were utilized to

form the interview questions for data collection. Once the interview questions were formed and reviewed we conducted the interviews in order to gain a better understanding of the research problem and its scope. Furthermore, in order to categorize and analyze the collected data, the *Grounded Theory Method (Glaser & Strauss, 1967)* was applied as described by *Martin & Turner (1986)*. The collected data was categorized in order to identify abstract concepts/categories regarding the research problem. The categorization of the collected data helped us in enhancing the current set of identified keywords and this finalized set was then later used to search for relevant literature.

Under the action planning phase this thesis provided a concrete set of good practices that software as a service organizations can use in order to maximize customer satisfaction. The good practices suggested by this thesis are based on the results of the literature review and the empirical study (data collection and data analysis).

A concrete step taken under the specifying learning phase of action research was to discuss the results of the empirical study with the company supervisor. Furthermore, the conclusions drawn from the thesis were also discussed with the company supervisor. Later on, based on one of the key challenges identified in this thesis the case company was able to start a new research thesis. Table 3 specifies the research methods used to answer each research question

Table 3. Research methods used for each RQ

Research Question	Research Method
What is the current state of small-scale agile development at the case company?	• Semi-structured interviews within the case company • Qualitative data analysis using Grounded Theory
What good practices can be used by the case company in small-scale agile development projects to maximize customer satisfaction?	• Literature Review • Action planning used to suggest good practices based on qualitative data analysis and literature review • Specifying learning used to discuss the good practices and future prospects with the company supervisor.

2.2 Literature Review

The literature review of this thesis focused on the Research Question: *What good practices can be used by the case company in small-scale agile development projects to maximize customer satisfaction?*

Through the literature review, this thesis tries to identify the best practices that are followed by small-scale agile projects in SaaS companies. In order to start the literature review, a set of relevant keywords was extracted from the research problem and questions. The literature review was conducted after the empirical study in order to gain clarity regarding the key concepts related to the research problem. Data collection and analysis done under the empirical study further helped in finalizing the keywords that encapsulated a more concise

meaning of the research problem. For this reason, the set of keywords were updated if, and when, needed based on the results of data analysis.

These identified keywords were used to query various online databases such as Google Scholar, IEEE Xplorer, and Springer. Each query resulted in numerous sources and there was a need to filter the results according to their relevance to the main research problem. In order to filter articles, the first step was based on the relevance of the article's abstract with the key concepts of the thesis.

After the initial selection, the shortlisted articles were explored further by in-depth reading. Furthermore, snowballing was also used to find more relevant sources for the research problem. Literature Review was focused on finding answers to the following areas:

- Understanding the basic concepts that were used to define the research problem
- Standard practices that have been defined in Academic Research

2.3 Empirical Study

2.3.1 Case Description

The case company offers supply chain solutions to various stores and retailers in the form of a software product and has over 700 employees with offices all across the world, including the United States and Finland. The company operates on a SaaS (Software as a Service) delivery model.

Customers are moved into the *continuous service phase* after the project has been set up and the customer environment has been operating in a stable manner for a while. The Continuous service phase is further divided into three main areas in the case company: *Service Delivery, Professional Services,* and *Customer Success*. *Service Delivery* focuses

on the contractual requirements for SaaS customers such as regular version upgrades of the software and non-billable support. *Professional Services* encompasses all the billable services that the case company provides outside of the SaaS contract such as any new *request* made by the customer to modify the existing behavior of their environment in order to improve or introduce new capabilities. These requests are treated as *small-scale agile development projects*. Such projects shall take less than 10 working days to complete and are invoiced based on time spent on development and validations. However, within the context of the case company, they are called “Minor Development Projects”. *Customer Success* focuses on the general well-being of the existing customers and ensuring that the company does not lose its customer base.

For the purpose of this thesis, we will be focusing on the small-scale agile development projects (or Minor Development Projects) and the impact of their delivery on customer satisfaction. The delivery of such projects has a direct correlation with the overall satisfaction of the customer towards the product. Employees belonging to the business and technical teams are involved throughout such projects in various capacities.

Currently, the existing material within the company highlights the process of such small-scale projects on an abstract level. However, little or no knowledge can be found on the good practices that software developers should follow in order to achieve customer success. The goal of this thesis is to highlight good practices that should be followed by software developers throughout the minor development process in order to ensure maximum customer satisfaction.

2.3.2 Research Process

Figure 4. explains the activities conducted as a part of the empirical study research process and also maps them to action research activities.

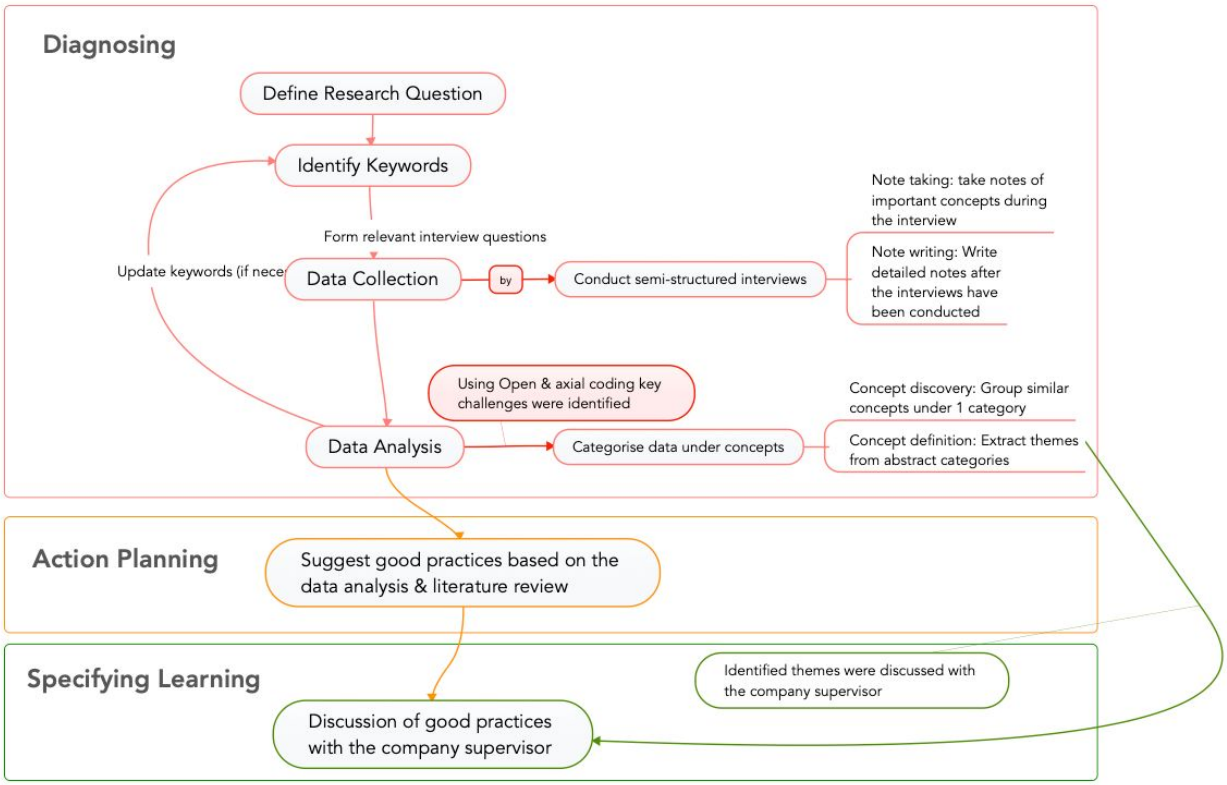

Figure 4. Overview of the empirical research process

As a part of the diagnosing phase we first identified the research problem and questions. Based on this we identified a set of keywords that define the scope of our research problem. semi-structured interviews were used as the main data collection process in this thesis. The collected data was then analyzed and evaluated using grounded theory as described by *Martin and Turner (1986)* in their research *Grounded Theory and Organizational Research.*

Martin & Turner (1986) divided the grounded theory method into three main phases that can be applied during organizational research. These three main phases are namely: *Note writing, Concept discovery and Concept definition (Martin and Turner, 1986)*. Note writing phase can be further be divided into two distinctive phases i.e. *Note taking* and *Note writing*. *Note taking* here refers to the activity of making concise and relatable notes while the interview is being conducted *(Martin and Turner, 1986)*. *Note writing* is the process of writing detailed notes from the interview recordings no later than a day after the interview has been conducted *(Martin and Turner, 1986)*. *Concept discovery and Concept definition* were used to analyze the collected data further along the thesis. *Concept discovery* is the process of grouping similar incidents from the raw data under abstract codes or categories. *Concept definition* involves extracting themes from the abstract categories based on the core attributes of each reported incident *(Martin and Turner, 1986)*. Data collection activity of empirical research utilized the concepts of note taking and writing. Data analysis was done by utilizing concept discovery and concept definition. Under the data analysis phase we focused on identifying the key challenges that small-scale agile development projects faced at the case company.

Once the data analysis was completed the results of the empirical study were then shared with the company supervisor in a meeting. The purpose of this was to understand which of the identified concepts from empirical study were of importance from the case company perspective. Furthermore, the results of the literature review were then used to suggest good practices to the challenges that were identified as a part of the data analysis. Towards the end these suggested good practices were again discussed with the company supervisor as a part of specifying learning.

2.3.3 Data Collection

The main data collection method used in this thesis was s*emi-structured in-depth interviews*. As per research, such interviews allow the researcher to understand the interviewees' perspective in a deeper manner and to answer a wider range of questions *(Dicicco and Crabtree, 2006)*, which further makes it a preferable data collection approach for this thesis. Figure 5 shows an overview of the data collection process

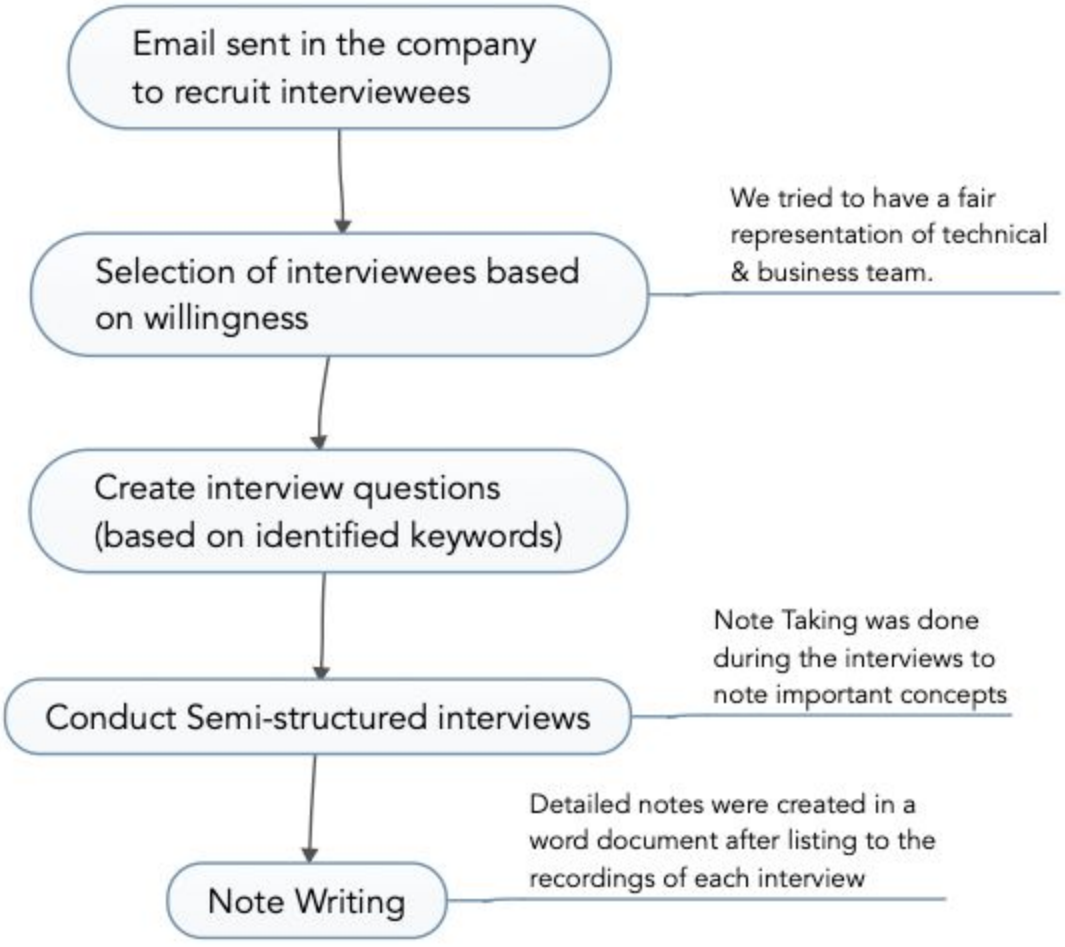

Figure 5. Overview of data collection process

An email, announcing the need to recruit interviewees for the thesis, was circulated among a few shortlisted employees from the business and technical teams that are involved in

small-scale agile development projects. Six employees were selected based on their willingness to participate in the research after the first circulation of the email.

There was a fair representation of technical and business teams involved in small-scale agile development projects as one of the main goals of the thesis was to analyze the current state of these projects at the case company. The roles of the interviewees within the case company are listed in Table 4.

Table 4. Roles of interviewees at the case company

Name	**Role**
Interviewee 1	Director, Technical Delivery
Interviewee 2	Customer Success Manager
Interviewee 3	Team Lead Customer Success Management
Interviewee 4	Technical Delivery Manager
Interviewee 5	Technical Consultant
Interviewee 6	Technical Consultant

Interview questions were carefully extracted and designed based on the main research question defined towards the beginning of the thesis and can be found in Appendix 1 of this thesis. Furthermore, the interview questions were structured according to specific categories. The interview questions were slightly modified for each interviewee based on their role within the scope of small-scale development projects at the Case Company.

All the interviews were conducted on-premise with the exception of one interview as the interviewee was from one of the country offices of the Case Company. All interviews were recorded, after taking permission from the interviewee, in order to enable us to write detailed notes of the interviews later on. The planned duration of the interview was 1 hour. However, in general, the interviews ranged from 40 minutes to a total of 60 minutes, based on how concisely the interviewee answered the questions. During the interview session, notes were taken in the form of keywords and key concepts that were discussed with the

interviewee. These notes were made with the purpose of making the process of *Note writing* easier *(Martin & Turner, 1986)*

Later on, the interviews were transcripted from the recordings as soon as possible and no later than the day of the interview itself. The discussions from each interview were transcripted in a separate word document under anonymous names in order to hide the identity of the interviewee. The interview notes were written in a structural way that followed the format in which the interview questions were categorized (as shown in Appendix 1).

2.3.4 Data Analysis

The data analysis phase was mostly designed to answer the first research questions: *What is the current state of small-scale agile development at the Case Company?* However, we were also able to suggest good practices to the case company based on the suggestions that were brought forward by the interviewees. Figure 6 gives an overview of the data analysis process.

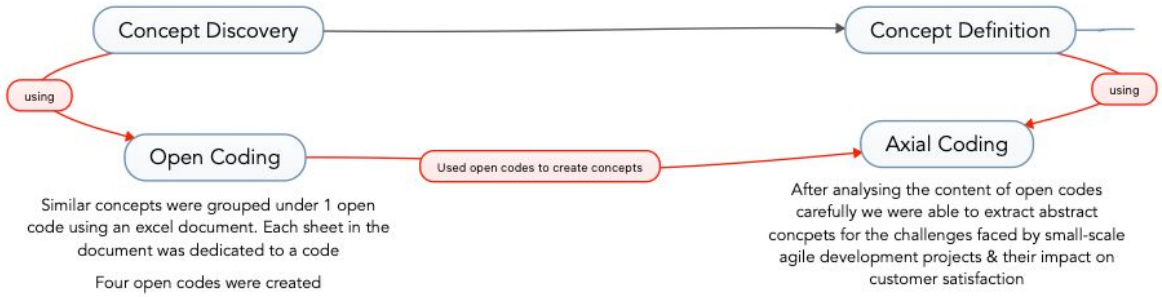

Figure 6. Overview of data analysis process.

In order to analyze and categorize the interview data, we utilized the core principles of grounded theory such as *open coding* and *axial coding* as defined by Muller M., (2014). Open coding can be defined as the description of a phenomenon, for example, the

description of the current situation at the workplace. Axial coding is used to group several open codes under a broader abstract concept *(Muller, 2014)*.

For the purpose of making the coding process structural, we utilized *concept discovery* and *level of abstraction* concepts provided by *Martin and Turner (1986) in their paper Grounded Theory and Organizational Research*. *Concept discovery* refers to the process of converting raw data into abstract concepts that relate to the phenomenon that is being studied. The idea is to create abstract concepts or labels and group relevant sets of data under them *(Martin and Turner, 1986)*. *The level of abstraction* provides a structural approach towards creating these concepts *(Martin and Turner, 1986)* . The main aim is to create a concept abstract enough so that each identified open code does not end up with its own unique concept. The other criteria is to make sure that the concept is explicitly related to the phenomenon under study *(Martin and Turner, 1986)*.

The raw data from the semi-structured interviews were first collected in a spreadsheet where it was divided into open codes that were explicitly referred to in the interview questions such as the current situation and bottlenecks (see Appendix 1). Open coding was done in an excel document with each excel sheet dedicated to one open code. Creating open codes helped in identifying relevant concepts during the axial coding process. Key sets of challenges were identified and grouped under abstract concepts as a part of axial coding. Axial coding was done using an excel document. Each sheet of the excel document was dedicated to one unique abstract concept. During the axial coding process we also further analyzed the relationship between the key challenges that were identified. Based on this analysis the challenges were further grouped according to the impact they had on customer satisfaction.,

3. Literature Review

The main aim of this section is to understand the relationship between small-scale agile development, Software as a Service organizations (Saas) and customer satisfaction. In order to better understand the relationship between small-scale agile development in SaaS organizations and customer satisfaction, it is important to first understand these concepts in isolation. The section is structured in a way to first give a definition of Software as a Service (SaaS) organization and how small-scale agile development is generally implemented in such organizations. This is followed by a brief overview of customer satisfaction and its importance in SaaS organizations. Lastly, this section analyzes the impact small-scale agile development projects have on customer satisfaction from the perspective of a SaaS company. Based on this knowledge the section will list a few good practices that can be used during a small-scale agile development project in order to maximize customer satisfaction. Later on this knowledge will be used to answer the second research question i.e. help create a set of good practices that maximize customer satisfaction and can be used in the case company.

3.1 Software as a Service Model

Software as a Service (SaaS) is a model where the software product, service or application is hosted on a cloud network. Users can access the SaaS product, with an annual usage fee, via a network such as the Internet *(Seethamraju, 2017)*. There are currently various SaaS products present in the software industry such as *enterprise resource planning* (ERP) and *customer relationship management* (CRM). The annual fee associated with the usage of SaaS products is determined by various factors such as *total number of users* and the functionality being used by the customer *(Loukis et al., 2019)*.

3.1.1 Service level agreements in SaaS

Service level agreement (SLA) is a contract between the customer and the SaaS provider that defines the *functional* and *non-functional* services that the customer is eligible for *(Gao et al., 2013)*. Design is of utmost importance in most products and SaaS is not an exception. *Aleem et. al (2018)*, in their research identified five key design factors that should be

incorporated in a SaaS product in order to ensure a well-designed product. In this thesis we will be discussing one of them in detail namely, Quality of Service (QoS) differentiation.

Customers, or users, of SaaS products have a strong demand for quality *(Gao et al., 2013)* and this creates a complexity in system design known as *Quality of Service differentiatio*n *(Krebs et al., 2012)*. Different users or customers might have varying quality demands and willingness to pay for the product. This difference of customer demands eventually leads the SaaS organizations to offer *different service classes* from which the customers can choose according to their requirement and willingness *(Aleem et al., 2018)*. Service classes can be easily explained with an example of *Netflix.* Netflix offers various packages with varying costs and users can select the package that suits them the most. These various packages are different service classes that Netflix offers to its users. Customers of a SaaS product or service belong to different service classes which implies they receive different levels of services and this is known as *Quality of Service differentiation (Krebs et al., 2012)*.

The *quality of service (QoS)* provided to each customer is defined by the *service level agreement (SLA)* that SaaS organizations negotiate with the customers *(Aleem et al., 2018)*. Each customer has different needs hence a different s*ervice level agreement* which implies that the *quality of service* for every customer will be different as well *(Aleem et al., 2018)*. The *quality of service* depends on the service class that the customer chooses *(Aleem et al., 2018)*.

3.1.2 Customer success, satisfaction & loyalty in SaaS

There are a few concepts that come under the umbrella of customer success such as *customer satisfaction* and *customer loyalty*. Customer satisfaction and loyalty are a result of the value that the customer feels they are receiving from the use of a product or service. Figure 7 summarizes the relationship between customer success, customer satisfaction and customer loyalty from the perspective of a SaaS company.

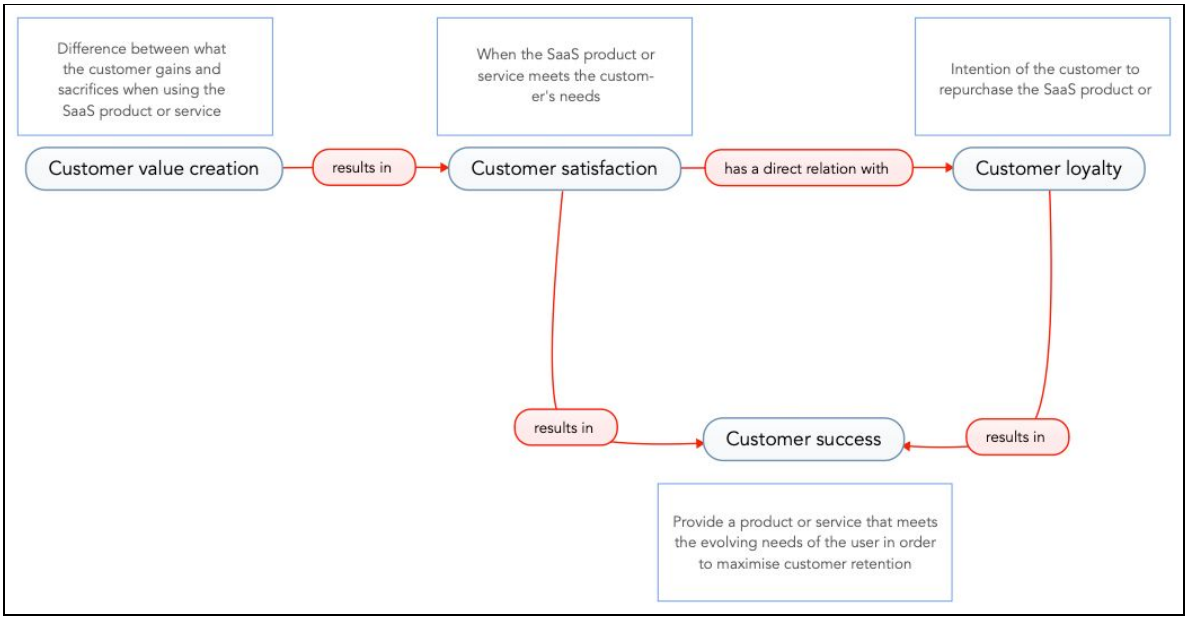

Figure 7. Relationship between customer success, customer satisfaction and customer loyalty

In this era, customers try to look for products that provide them value *(Grönroos, 2007)*. Customer value can be understood as the difference between what the customer gains from the product or service and the sacrifices the customer has to make in order to use the product *(Kauppinen et. al, 2009)*. Customer gains refers to the quality of the product and the benefits the customer receives by using the product *(Normann et. al, 1993)*. The sacrifices that customers have to make can be the monetary cost of buying the service and maintaining cooperation with the service provider i.e. the company *(Normann et. al, 1993)*.

Once the customer feels that the use of a product or service is providing them value it results in customer satisfaction. *Oliver (2010)* in his book defined customer satisfaction as a customer's perception of whether a product or service meets the customer's needs or not. This can result either from the reduction of customer's already existing problems or by providing some additional unexpected benefits *(Oliver, 2010)*. Customer satisfaction seems to have a direct relationship with customer loyalty, positive word of mouth and competitive position for the service provider or the company *(Bearden et. al, 1983)*. Customer loyalty, in SaaS, can be seen as the intention of the customer to repurchase the product or service *(Fornell, 1992)* as SaaS is subscription based model *(Seethamraju, 2017)*. Customer loyalty ensures recurring revenue for SaaS business *(Mehta et al., 2016)*. Word of mouth refers to the conversation regarding the evaluation of a product between an existing customer and potential customers *(Anderson, 1998)*.

At its very heart customer success can be seen as the practice of focusing on the experience of a customer with the product or service being offered to them in order to maximize customer retention. Achieving customer success requires having a constant relationship with the customer in order to inquire about their needs and adapt the product or service as per the evolving customer needs *(Mehta et. al, 2016)*. Achieving customer success in a SaaS organization ensures customer retention which is important for the overall revenue in SaaS. Furthermore, word of mouth plays an important part here as well as it brings *second-order revenue* for a SaaS company *(Mehta et. al, 2016)*. *Mehta et. al (2016)* defines

second-order revenue as the revenue that comes from gaining a new customer based on an existing customer's reference or positive review.

3.2 Small-scale agile development

Software industry has now moved towards agile software development due to its ability to deliver good quality software to the customer quickly. The 4 core values of agile as listed in the agile manifesto *(Fowler and Highsmith, 2001)* are:

- Value individuals and interactions over processes
- Value working software over documentation
- Value customer collaboration over contract negotiation
- Adapt to change over following a plan

Figure 8 summarizes the core values of agile development as defined by *Fowler and Highsmith (2001)*

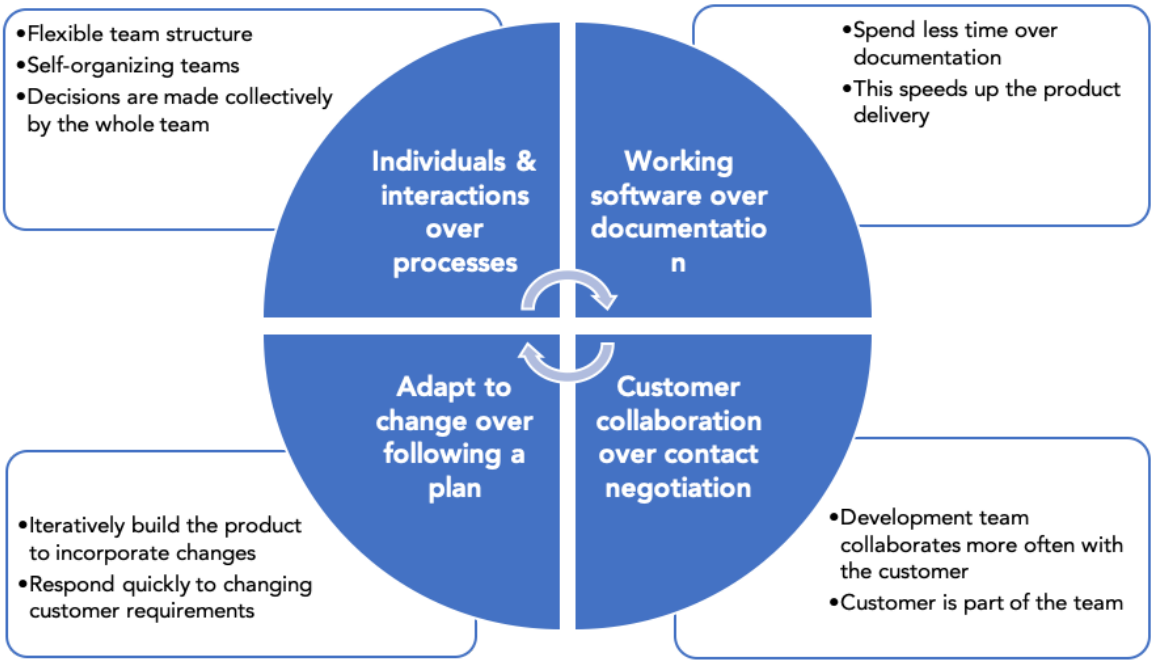

Figure 8. Four core values of agile development

Apart from just quick delivery, agile development methods also ensure customer satisfaction by incorporating rapid customer feedback in the development process *(Zhang and Dorn, 2011)*. Furthermore, agile methodologies cater to changing needs of the customer and believe in delivering value to the customer *(Nerur and Balijepally, 2007)*. The primary reasons behind the emergence of agile software development was to address the changing customer requirements and business environments along with evolving technology trends *(Drury-grogan et. al, 2017)*.

Agile methodologies are theoretically built for small, self-organizing teams *(Lee and Yong, 2013)*. Software projects can generally be categorized as small, medium or large. According to Lee and Yong (2013), factors that can be used to categorize a project as small can be: the duration of a project, number of team members involved, the scope of the project and the technical impact of the project. Small projects usually last less than six months, have ten or fewer team members along with a concisely defined scope *(Lee and Yong, 2013)*.

A key feature that SaaS organizations provide to their customers is continuous improvement of the product i.e. introducing new value-adding features to the product in order to create more business value for the customer *(Nerur and Balijepally, 2007)*. One key benefit associated with use of a SaaS model is known as: *Operational benefits*. By using a common infrastructure when providing the product to multiple clients, organizations reduce the overall *operational cost* and also increase the *quality of customer support (Loukis et al., 2019)*. Software organizations are widely moving towards the SaaS model as it differs from the 'traditional' on-premise model and allows them to cater the needs of multiple users by using a service that is hosted on a single location *(Loukis et al., 2019)*. It enables the organization to share their current service across multiple customers *(Benlian and Hess, 2011)*.

3.4 Challenges and good practices of agile development

3.4.1 Geographically distributed team challenges in agile development

Fitriani et. al (2016) did a systematic literature review to identify the most frequent challenges associated with agile software development. They identified a list of 30 challenges after reviewing 20 studies. The challenges that were encountered the most in literature, an approximate occurrence frequency of 45%, were *Team management* and *Distributed team*. One of the core principles of agile development prioritizes individuals and interactions over processes *(Fowler and Highsmith, 2001)*. This prioritization can sometimes lead to difficulties in team management *(Fitriani et. al, 2016)*. *Team management* refers to coordination within the team and requires special attention to ensure successful software development *(Fitriani et. al, 2016)*.

Team management can become more challenging if the team is distributed globally. This is known as "geographically distributed agile development" or GDAD *(Ibrahim et. al, 2016)*. GDAD refers to a situation in which team members of an agile software development team are working together to accomplish a project from separate geographic locations. The team members can be distributed within the same country or can be globally around the world across different time-zones or countries *(Ibrahim et. al, 2016)*.

GDAD team design offers the opportunity to hire talented developers from across the world and reduce the production cost *(Ibrahim et. al, 2016)*. GDAD, however, also poses challenges for agile development and the biggest of these challenges is *poor communication (Herbsleb and Moitra, 2001)*. *Poor communication* refers to the challenge of delivering incomplete, inaccurate or inadequate information within the team *(Herbsleb and Moitra, 2001)*.

Communication is important in a software project because of two reasons. Firstly, to make sure that the team understands the customer's requirement clearly. Secondly, to determine the responsibilities within the team and to make sure that project status is being updated correctly *(Herbsleb and Moitra, 2001)*. Table 5 summarizes communication challenges that arise among geographically distributed teams.

Table 5. Communication challenges in GDAD teams

Challenge	**Description**
Distance differences	This refers to the difference in time-zone and the physical distance between team members. Due to these differences it can be difficult to arrange team meetings and can be costly to arrange face-to-face meetings. This eventually leads to a reduced amount of trust between the GDAD team members.
Human factors	This refers to the differences in culture and language across the GDAD team members according to their location or country. These differences can lead to a number of challenges such as misunderstanding due to language differences, lack of mutual understanding and poor coordination due to different holidays in different locations.

Distance differences occur in two different forms such as the physical difference between the team members and the time-zone differences *(Ibrahim et. al., 2016)*. Due to the difference in time-zones it can be difficult to arrange team meetings within the standard local working hours of all the time-zones involved. Due to the physical distance between the team members it can be costly to arrange face-to-face meetings for the GDAD team. Team members often have ad hoc communication in scenarios where they need to discuss an aspect of the project and clarify the details. This can be difficult to manage in GDAD teams due to distance differences and causes significant delays *(Sindhgatta et. al., 2011)*. These delays in communication can make team members frustrated due to feeling of

"missing out" on information and also lose track of work *(Holmström et. al, 2006)*. Less face-to-face meetings also reduce the chance to hold informal gatherings between the team which leads to a reduced amount of trust between the GDAD team *(Holmström et. al, 2006)*. Furthermore, this can minimize the overall efficiency and effectiveness of team communication *(Ibrahim et. al., 2016)*.

Human factors refers to the fact that a GDAD team comprises team members from different cultures and languages. This means that the norms, values, language and style of communication of each team member will be different according to their location or country *(Ibrahim et. al., 2016)*. The level of trust expressed by each team member is dependent on the culture of the location they are coming from *(A. Boden et. al, 2007)*. These differences can lead to a number of challenges such as misunderstanding due to language differences, lack of mutual understanding *(Holmström et. al, 2006)* and poor coordination due to different holidays in different locations *(P.L. Bannerman et. al, 2012)*. Furthermore, some team members might be more silent in comparison to others based on their cultural differences and this can lead to ineffective coordination over video conference meetings *(A. Boden et. al, 2007)*. Figure 9 summarizes the challenges that arise due to distance differences and human factors.

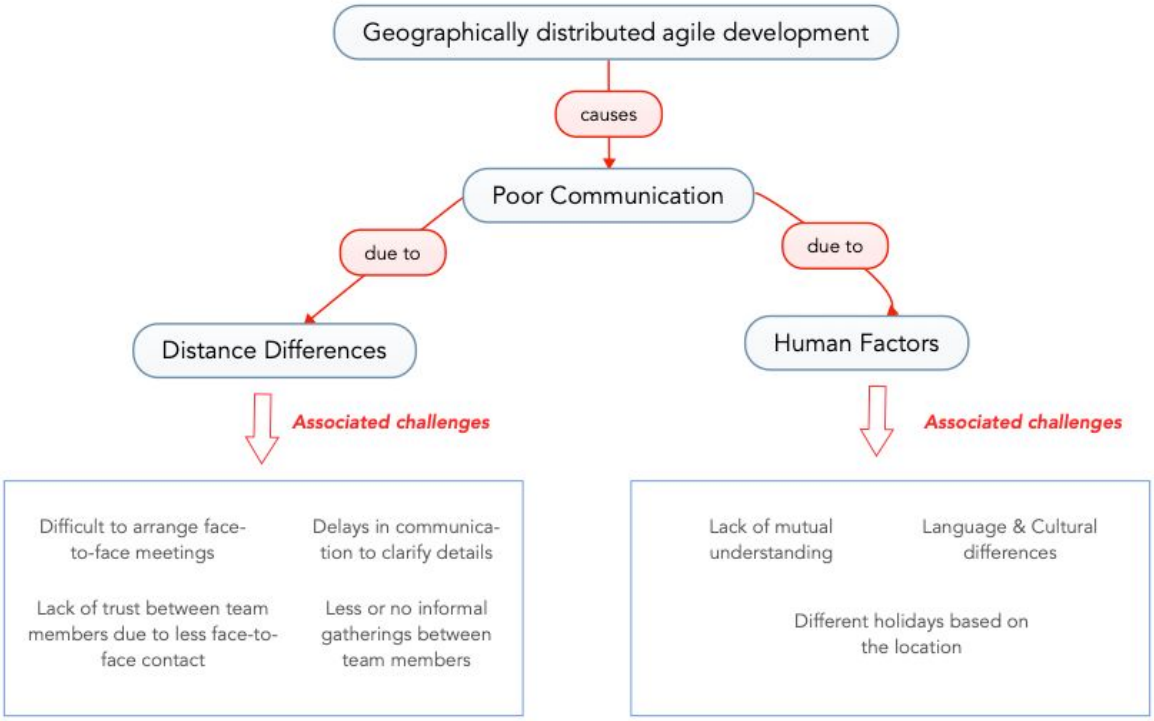

Figure 9. Challenges resulting from distance differences

Figure 10 summarizes good practices that can be used to reduce the impact of distance differences and make communication more effective.

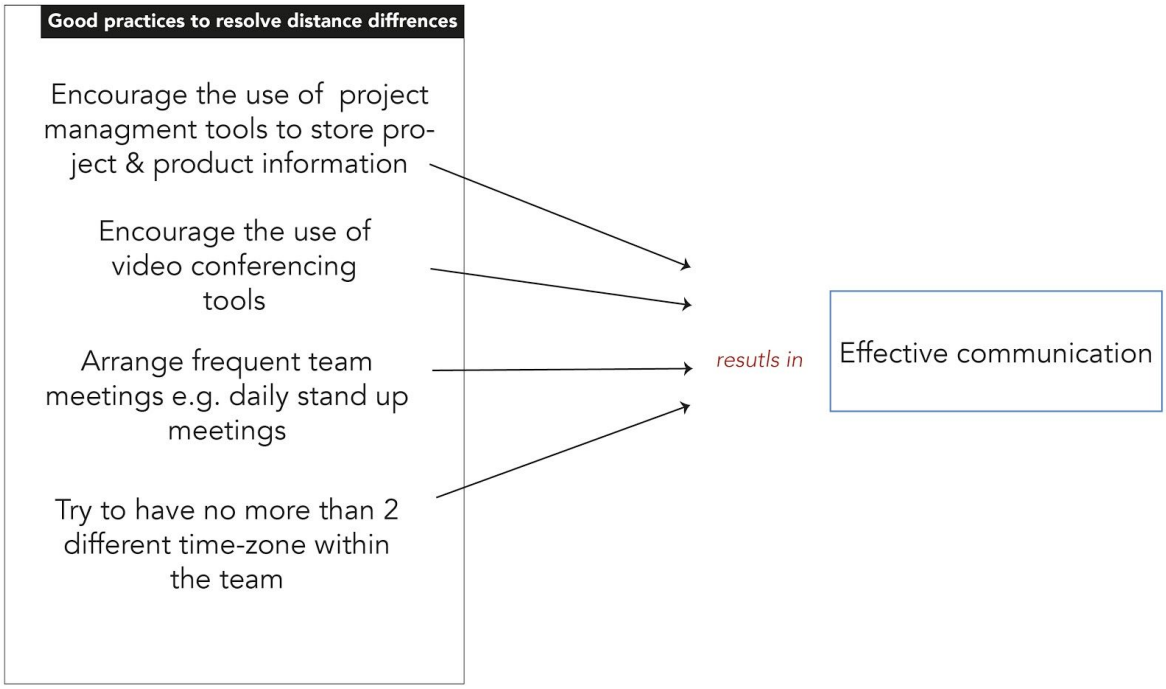

Figure 10. Good practices to resolve distance differences

In order to ensure visibility of a GDAD project it is necessary that process and product information is stored in a knowledge base that is easily accessible to all team members *(Layman et al., 2006)*. *Layman et al. (2010)* recommended for GDAD teams to use globally-available project management tools to record and monitor the current status of the project. This practice provides visibility to the development team as well as the customer *(Layman et al., 2006)*.

To compensate for distance differences it is important to establish communication mediums such as synchronous communication and asynchronous communication *(Green et al., 2010)*. Synchronous communication mediums refers to video conferencing and teleconferencing or instant messaging *(Green et al., 2010)*. Asynchronous communication mediums refer to emails and documentation *(Green et al., 2010)*. *Green et al. (2010)* highlighted that most successful GDAD teams have rich synchronous communication mediums especially at the beginning of the project. Synchronous communication resolves the issue of delayed ad hoc communication in GDAD teams *(Sindhgatta et al., 2011)* and also allows the team members to read each other's body language *(Dorairaj et al., 2011)*. However, in some cases establishing synchronous communication can be costly and in these cases the team should then utilize asynchronous mediums in a more efficient manner *(Layman et al., 2006).*

Furthermore, during the project it is important to hold demonstration meetings to give the team an overview of the current state of the project *(Green et. al., 2010)*. This practice helps in aligning the vision of the team *(Green et. al., 2010)*. *Dorairaj et al. (2011)* also recommended that the GDAD team should have formal meetings such as weekly meetings and daily stand-up meetings. These meetings make it easier for the team members to respond to change and gain a mutual understanding of the project *(Dorairaj et al., 2011)*.

Sindghatta et.al. (2011) recommended that GDAD teams should not be distributed in more than two different time-zones. Ideally, there should be a 2-3 hour overlap of working hours between the team members *(Dorairaj et al., 2011)*. Furthermore, the GDAD team's work environment should be flexible as team members might need to communicate with each other outside of working hours due to time-zone differences *(Sindhgatta et al., 2011)*. Figure 11 summarizes the good practices that can be used to reduce the impact of human factors and increase trust among team members.

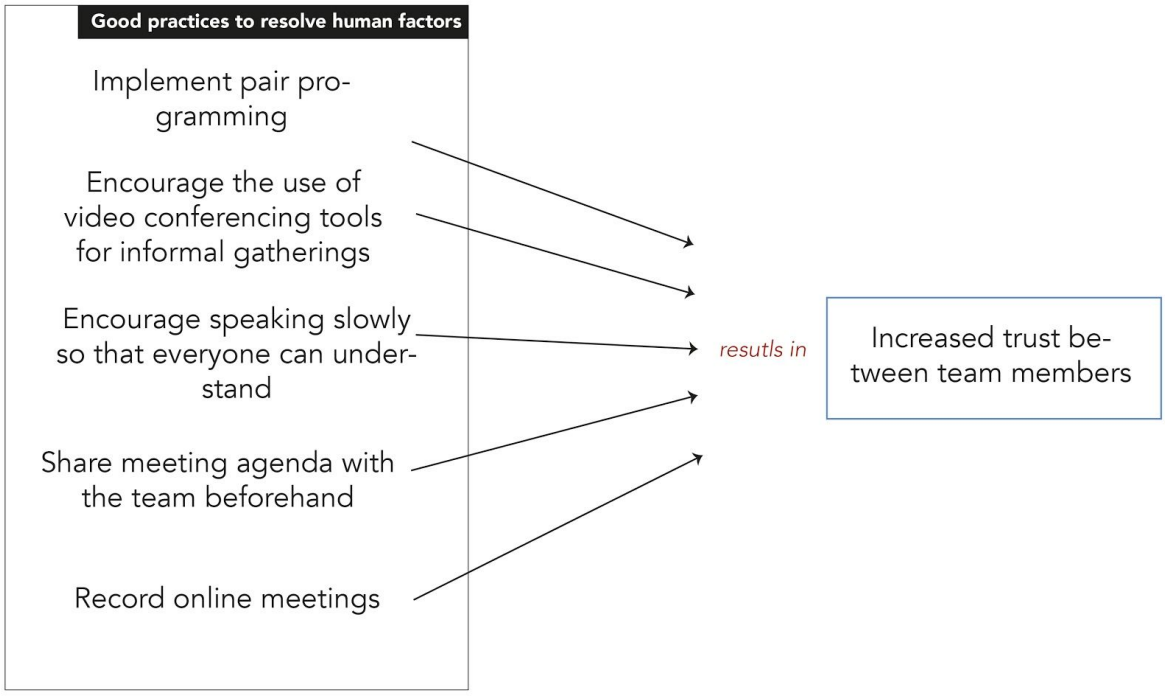

Figure 11. Good practices to resolve human factors

In a GDAD team environment it can be difficult to build a good relationship between the team members due to distance differences and sociocultural differences. *Dorairaj et al. (2011)* pointed out that GDAD team members work more effectively together once trust has been established between them. For this reason, *Holmström et al, (2006)* suggested to implement pair programming in a GDAD team in order to create a collective ownership of the project. In practice this means that two developers work together on the design of the same software *(Begel et al., 2008)*. One of the two developers is known as the *driver* and is responsible for actively writing the code or recording the design of the software *(Begel et*

al., 2008). The second developer is known as the *navigator* and is responsible for actively going through the work done by the driver in order to identify defects *(Begel et al., 2008)*.

A few benefits of pair programming that *Holmström et. al, (2006)* deduced from their research are high code quality and fresh viewpoint while testing. *Begel et al. (2008)* also identified in their research that pair programming resulted in less number of software. *Hulkko and Abrahamsson (2005)* also reported that pair programming helps in finding defects in the code. Another important benefit of pair programming is that it helps the developers in creating a common understanding of the software product that they are developing and its code *(Begel et al., 2008)*. *Hulkko and Abrahamsson (2005)* concluded from their research that pair programming helps in gaining understanding of the project for both the developers especially at the beginning. *Lui and Chan (2006)* highlighted in their research that pair programmers work efficiently when they are faced with "challenging programming problems" instead of straightforward problems. Challenging programming problems here refer to problems that require sophisticated and non-straight-forward algorithms to solve *(Lui and Chan, 2006)*.

Another factor that contributes towards trust building in the GDAD team is informal conversations such as casual chat between team members via video conferencing or teleconferencing.. *Dorairaj et al. (2011)* pointed out that it is important for the GDAD team to encourage informal conversations between the team members in order to build a strong relationship. Informal conversations also increases knowledge transfer between the team members *(Dorairaj et al., 2011)*. Furthermore, using a globally-available tool to monitor project progress distributed team members can easily participate in the whole process *(Layman et. al., 2006)*. This eventually leads to a feeling of working together for the team members and generates trust between them *(Holmström et. al, 2006)*.

In order to resolve the language barriers Layman et al. (2006) suggested to appoint a role within the team who works with all team members and can preferably speak all the involved languages. *Dorairaj et al., (2011)* recommended to ensure that team members

speak slowly and clearly so that everyone can understand them. In order to effectively communicate in a meeting it is also recommended to share the meeting's agenda with the GDAD team beforehand *(Dorairaj et al., 2011)*. And when possible the online meetings should also be recorded so that people who are unable to attend a meeting can later on listen to the recording *(Dorairaj et al., 2011)*. Table 6 summarizes a few good practices that can be used to minimize the negative impact of communication challenges.

Table 6. Good practice to minimize communication challenges

Good practices to minimize communication challenges	Literature Reference
Encourage the use of globally-available project management tools	*Layman et al. (2010)*
Encourage the use of synchronous communication mediums such as teleconferencing and video conferencing tools.	*Green et al., (2010), Sindhgatta et al., (2011), Dorairaj et al., (2011)*
Encourage/arrange frequent team meetings	*Green et al., (2010), Dorairaj et al., (2011)*
Encourage having a project team that is divided into no more than 2 different time-zones.	*Sindhgatta et al., (2011), Dorairaj et al., (2011)*
Encourage pair programming in order to create collective ownership and understanding of the project	*Holmström et al, (2006), Begel et al. (2008), Hulkko and Abrahamsson (2005), Lui and Chan (2006)*
Encourage and establish informal conversations by arranging online meetings	*Dorairaj et al., (2011), Layman et al. (2010),Holmström et al, (2006)*
Encourage speaking slowly so that everyone can understand each other during a team meeting.	*Dorairaj et al., (2011)*
Encourage sharing the agenda of the meeting beforehand	*Dorairaj et al., (2011)*
Record online meetings so that absent team members can stay up-to-date on the progress of the project.	*Dorairaj et al., (2011)*

3.4.2 Documentation challenges in agile development

The agile manifesto clearly states the importance of a working software over comprehensive documentation *(Fowler and Highsmith, 2001)*. Customer is the main source of requirements in agile development and project success is directly tied to customer involvement *(Inayat et al., 2015)*. Agile methods heavily rely on frequent collaboration with the customer in the form face-to-face communication or onsite available customer instead of comprehensive documentation *(Inayat et al., 2015)*.

Customers are an integral part of agile development and need to be involved in the whole development process *(Ibrahim et. al., 2016)*. It is important that no project information is hidden from the customer *(M. Korkala et. al, 2010)*. Agile methods rely on the development team discussing the project requirements/specifications with the customer back and forth throughout the project *(Ramesh et al., 2010)*. This is also known as customer communication *(Ibrahim et. al., 2016)*. However, customers might not always be available or willing to answer questions for the development team. Two main factors that impact the quality of customer communication are *customer availability* and *the quality of the relationship between the two parties (Cao and Ramesh, 2008)*. Table 7 gives a summary of the challenges that arise due to minimal documentation.

Table 7. Minimal documentation challenges in agile development

Challenge	Description
Misunderstood customer requirements	It can get difficult to communicate with the customer due to factors such as customer unavailability and physical distance or time-zone difference between the customer & the agile development team. This lack of communication with the customer can result in low quality documentation of customer's requirements.
Poor relationship between the customer & the agile development team	Creating trust between the customer and the agile development team can be difficult due to the physical distance between the two parties. Also, customers might not be comfortable working with the agile way of documentation & might not share relevant information with the agile development team. Hence, the development team might not be able to understand the customer's requirements due to their poor relationship.
Poor communication within the agile development team	The agile way of minimal documentation might not always cover all the aspects of the project especially related to the customer's domain. Hence, it can get difficult to convey information regarding the project to new team members of an agile development team.

Cao and Ramesh (2008) pointed out in their research that it can be difficult to achieve on-site customer representation in most projects. *Pichler et. al. (2006)*, also pointed out in their empirical research that continuous customer availability is a challenge for agile projects. Discussing the project scope and requirements with the customer that are in a different time-zone or geographical location can be very cumbersome *(Ibrahim et. al., 2016)*. It can be difficult to arrange face-to-face meetings if the development team and the

customer are on different geographic locations or in different timezones. In such cases having minimal documentation can be inefficient *(Inayat et al., 2015)* and there is always a risk that the requirements are not correct or are inadequately developed *(Cao and Ramesh, 2008)*. Furthermore, this can also lead to the developers misunderstanding the customer's requirements. In such scenarios the developers might take critical decisions regarding the project based on their past experience or make an educated guess about the customer's need *(M. Korkala et. al, 2010)*.

The *relationship between the customer and the development team* is another factor that influences the quality of customer communication. It can be difficult to establish trust between the customer and the development team especially towards the early phases of the project *(Cao and Ramesh, 2008)*. Establishing trust between the customer and the development team becomes more difficult in a geographically distributed agile development team due to lack of face-to-face communication *(L. Layman et. al, 2006)*. Furthermore, customers don't always have a proper understanding of the agile processes *(Cao and Ramesh, 2008)*. Some customers are used to the traditional documentation style and might not be comfortable using the agile way of minimal documentation. In such cases it can be difficult to get relevant information from the customer *(Cao and Ramesh, 2008)* and this again leads to the developers making design decisions based on their own understanding of the customer's requirement *(M. Korkala et. al, 2010)*. Hence, minimal documentation is one of the most vital challenges that agile development teams face *(Inayat et al., 2015)*

Another challenge related to minimal documentation is that of conveying requirement changes to novice developers or new members of the development team *(Selic, 2009)*. It might not be possible to accommodate all team members in a meeting with the customer due to various reasons such as less space in the office or geographically distributed team members. In such cases, relying on minimal documentation can be insufficient to relay information within the development team. It can lead to a lack of understanding of the

requirements for novice developers *(Inayat et al., 2015)*. Table 8 summarizes a few good practices that can be used to create concise documentation of customer requirements.

Table 8.Good practices to resolve documentation challenges in agile development

Good practices to resolve documentation challenges	**Literature Reference**
Encourage the creation of concise user stories in place of heavy-weight documentation. This will help in creating a shared understanding of the requirements.	*Cao and Ramesh (2008), (Cohn, 2004), Inayat et al. (2015), Ramesh et. al. (2010),*
Encourage the customer to define the priority of the user stories. This will help in avoiding misunderstandings regarding the prioritization.	*Inayat et al. (2015)*
Ensure 100% visibility for the customer and the team by making project documentation accessible to everyone.	Layman et. al (2010)
Encourage the use of a standard template for defining user stories. The template itself does not matter as long as the whole team is using the same one. This will make it easier for new members to understand the requirements.	*Wake (2003)*, *Lucassen et. al. (2016)*
Encourage direct interaction between the customer and the development team.	*Cao and Ramesh (2008), Ramesh et. al. (2010),*

Problems arising from minimal documentation can be resolved to some extent by using concise user stories for documentation. Most agile development teams tend to create concise user stories as a way to define high-level customer requirements instead of creating heavy-weight documentation *(Cao and Ramesh, 2008)*. The primary goal of creating user

stories is to define customer requirements in an understandable manner by using the template

"As a *(type of user)*, I want *(goal)*, so that *(benefit or* why *is the change needed)*" *(Cohn, 2004)*.

User stories require input from the customers in order to clearly define their requirements *(Lucassen et. al., 2016)*. Customers are needed to specify the goal and the benefit behind implementing a requirement. The *why* part of a user story helps in removing ambiguities related to the requirements. The requirements are then refined in each iteration and the customer is responsible for prioritizing the requirements of the project *(Inayat et al., 2015)*. This process of back and forth communication between the customer and the development team helps in creating a shared understanding of the customer's expectation *(Lucassen et. al., 2016)* . Furthermore, having a common understanding of the customer's requirements eventually results in less number of defects or post-development changes requested by the customer *(Ramesh et. al., 2010)*. The primary reason being that the developers are able to identify the needs of the customer and this enables them to create the *right* software *(Lucassen et. al., 2016)*.

Lucassen et. al. (2016), pointed out that practitioners of user stories reported an increase in the quality of their work and overall productivity. Layman et. al (2010) suggested that all the project documentation should be accessible to both the customer and the development team. In order to maintain visibility throughout the project each user story should indicate the effort required to implement it and the current status of the story *(Layman et. al., 2010)*.

Implementing a standard structure or template for defining user stories within the development team can also be beneficial. The benefit of having a template is that it eventually improves the overall productivity of the team and the quality of the user stories *(Lucassen et. al., 2016)*. *Lucassen et. al. (2016)* highlighted the fact that the template itself does not matter as long as the whole team is using the same template. The usage of a common template ensures that the whole team is working in the same format and has a common understanding of the template. Even though the template itself does not matter, it

is important to put in place a quality criteria for the user stories. One such quality framework for user stories is known as INVEST *(Wake, 2003)* and is the most well-known framework *(Lucassen et. al., 2016)*. INVEST is an acronym that stands for the qualities a user story should have: Independent, Negotiable, Valuable, Estimable, Small and Testable *(Wake, 2003)*.

Another framework found in the literature regarding the quality of user stories is: Quality User Story (QUS) framework *(Lucassen et al., 2015)*. QUS framework divides the quality criteria of a user story into three concepts such as: *Syntactic, Semantic* and *Pragmatic (Lucassen et al., 2015).* These three concepts are further explained using a concrete list of characteristics. Table 9 provides a brief description of these concepts, the key characteristics that are a part of each concept and their description.

Table 9. Description of quality user stories framework concepts

Concept	Key characteristics	Description
Syntactic	● Atomic ● Minimal ● Well-formed	***Atomic:*** A user story should only discuss one feature. ***Minimal & Well-formed:*** A user story should define the type of user, what needs to be done and why is it needed. Any further information should be stored as additional notes.
Semantic	● Conflict-free ● Problem-oriented ● Unambiguous	***Conflict-free:*** A user story should not negate another user story within the same project. ***Problem-oriented:*** A user story should focus on the problem instead of the solution. ***Unambiguous:*** A user story should not be ambiguous internally and the relationship with other user stories should also be clear
Pragmatic	● Explicit dependencies ● Independent ● Uniform ● Unique	***Explicit dependencies:*** The link between user stories that are dependent upon each other should be clear. ***Independent:*** A user story should be schedulable and implementable in any order. ***Uniform:*** All user stories should follow the same format. ***Unique****: A user story should be unique and should not be duplicate of another user story.*

One of the most important characteristics discussed by *Lucassen et al. (2015)* in their research is the atomicity of the user story. Atomicity refers to the concept that a user story should only discuss one feature at a time *(Lucassen et al., 2015)*. Most importantly, a user

story should not negate another user story in the set. Furthermore, user stories should focus on the problem instead of the solution. *Lucassen et al. (2015)* further states that the link between user stories should be documented by using a traceability matrix. It is also important that a user story should be independent of other user stories i.e. it should be possible to implement a user story independently *(Lucassen et al., 2015)*.

Cao and Ramesh (2008) pointed out that the success rate of requirement specifications via user stories was highly dependent on the quality of interaction between the customer and the agile development team also known as *customer communication (Ibrahim et. al., 2016)*. As per Cao and Ramesh (2008), in the majority of the projects they studied, the project managers were representing the customer. This implies that project managers usually act as a buffer between the customer and the development team. This approach has a negative impact on customer communication as it limits customer interaction which further increases the risks of misunderstood requirements *(Ramesh et. al., 2010)*. Even though creating concise user stories solves the documentation challenge of agile development, the success of this approach is highly dependent on the user-developer interactions. It is important to eliminate buffers between the customers and the developers while creating user stories *(Ramesh et. al., 2010)*.

3.4.3 Software maintenance challenges in agile development

Another core principle of agile development is adapting to change. Agile development focuses on the adaptability and responsiveness to change when developing a software product. Customer's needs are changing constantly and accommodating new customer requirements is the key to customer satisfaction *(Cao and Ramesh, 2008)*. There are two kinds of changes that are often needed in agile development projects as pointed out by *Cao and Ramesh (2008)*:

1. Add or drop features from the existing software
2. Update an already implemented feature

Agile methods advocate less documentation *(Fowler and Highsmith, 2001)* and the code is considered to be a documentation of the system behaviour itself. However, with changing customer requirements and the addition of new features, the code base starts to become huge and complex. As per Cao and Ramesh (2008), the software architecture that the development team selects towards the beginning of the project becomes *legacy* or inadequate as new requirements emerge. As the code base starts to get complex due to changing/new requirements it can get difficult to use it as a source of understanding system behaviour. This lack of comprehension of the system behavior can lead to a decrease in the teams productivity and can also eventually affect the product quality *(Hanssen et al., 2009)*.

One serious threat that complex code-base poses is that developers prefer not to touch it as it is hard to understand and requires a substantial effort *(Selic, 2009 and Hanssen et al., 2009)*. Due to this developers sometimes prefer to create their own code instead of reusing an already existing code. This further adds to the complexity of the code base and creates a vicious cycle *(Hanssen et al., 2009)*. Furthermore, making small changes to a complex code base slows down the overall development process as developers need to modify code segments in one or more locations. This increases the risk of potential errors as the developers might overlook a code segment that needs to be modified *(Hanssen et al.,*

2009). Moreover, having a complex or legacy code base makes the learning curve of a new developer steeper as understanding the code base becomes difficult due to its overall complexity *(Hanssen et al., 2009)*. This means that the more experienced developers will have to closely follow up the progress of new developers for a longer period of time. This can cause a possible delay in their own tasks and can also slow down the overall process itself. Figure 12 explains the challenges associated with a complex code base in a comprehensive manner.

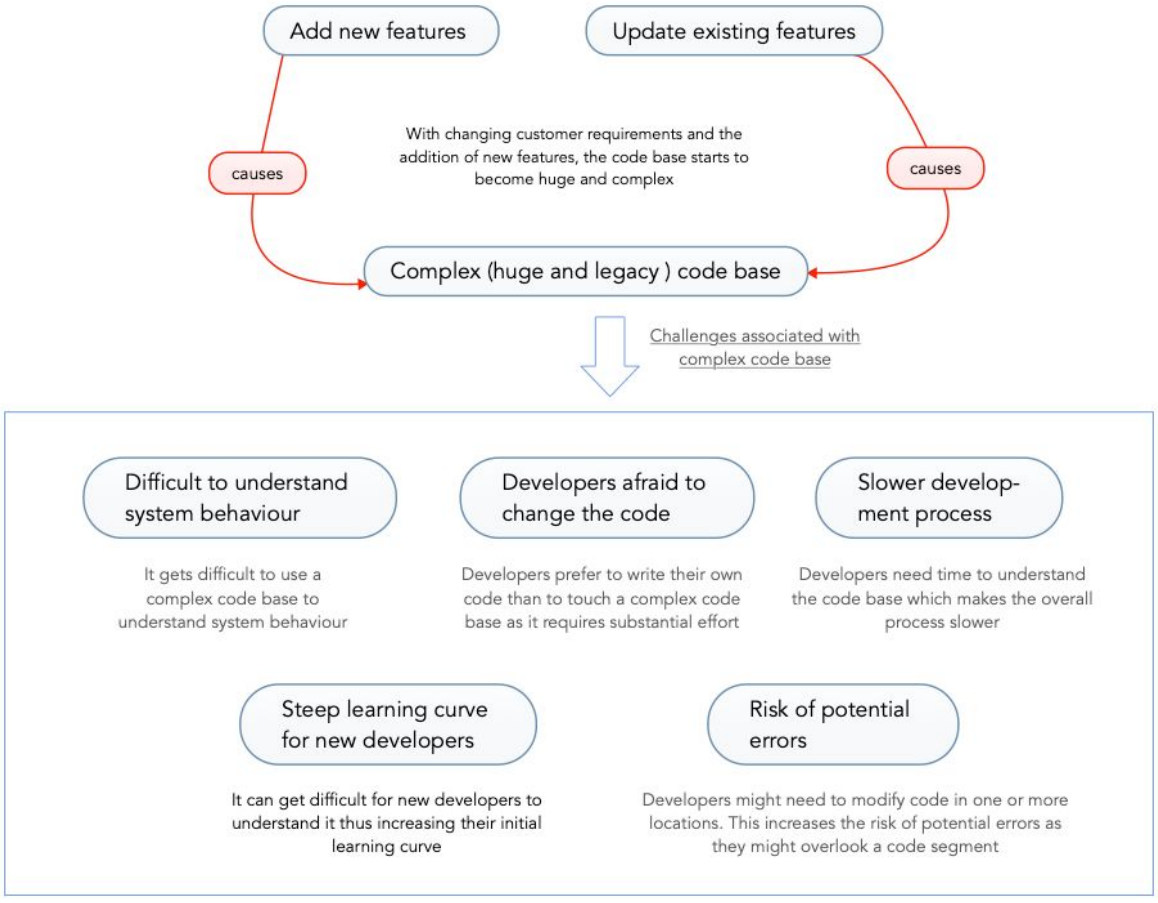

Figure 12. Challenges resulting from complex code base

Majority agile methods define best practices for the development process from the beginning of the project till the release. Best practices regarding post-release maintenance are usually not defined in enough detail *(Hanssen et al., 2009)*. One preferred post-release

maintenance activity is *code refactoring*. Code refactoring is an activity to modify and redevelop the existing code structure in order to make it more understandable and easy to change *(Inayat et al., 2015)*. Refactoring is an ongoing activity in agile development. If refactoring is stopped or avoided during agile development the code base can become legacy *(Ramesh et al., 2010)* which can make adding new features later on in the project costly and cumbersome *(Inayat et al., 2015)*. Table 10 summarizes a few good practices that can be used to create concise documentation of customer requirements.

Table 10. Good practices to resolve software maintenance issues in agile development

Good practices to resolve software maintenance challenges	**Literature Reference**
Encourage code refactoring on a regular basis.	*Cao and Ramesh (2008)*
Encourage the developers to analyze risk factors before providing a work estimation	*Hanssen et al. (2009)*
Encourage pair programming to deepen the understanding of the code.	*(Deursen, 2001)*

Code refactoring might not always be the solution for managing inadequate architecture resulting from changing requirements. Refactoring of the code depends on several factors such as the experience of the developer and deadline of the project. It might not always be possible to refactor the code due to a less experienced developer working on the project or due to a tight schedule to deliver the project *(Cao and Ramesh, 2008)*. Furthermore, Ramesh et al. (2010) also pointed out that code refactoring is not always a viable solution if the code implementation has become legacy. Refactoring legacy code implementation can be expensive when compared to rewriting the entire code again. Code refactoring adds a significant cost to the project and the project team might not always have the budget to accommodate it *(Cao and Ramesh, 2008)*. However, if refactoring is stopped or avoided

during agile development the code base can become legacy *(Ramesh et al., 2010)* which can make adding new features later on in the project costly and cumbersome *(Inayat et al., 2015).*

One good approach to get around the delays that complex code base introduces can be to conduct a thorough complexity analysis of the task at hand before providing work estimations *(Hanssen et al., 2009)*. Planning the work iterations while keeping in mind the complexity of the task ensures that the development team is aware of the risks and will plan accordingly. *Hanssen et al. (2009),* suggested the use of planning techniques such as planning poker when creating time estimations for tasks. Furthermore, pair programming is another technique that can be used to deepen the understanding of code for less experienced or new developers *(Deursen, 2001)*. The pairs can be created by looking at the experience of the developers in the team in order to make sure that new or less experienced developers are paired with new members. This will provide the new developers in the team with adequate support when working on a complex or legacy code base.

3.5 Summary of good practice in agile development

Three main challenges that agile development projects face can be categorized as follows:

1. Poor communication resulting due to a geographically distributed team
2. Incomplete or poor documentation of customer requirements
3. Software maintenance challenges

Literature suggests a few good practices that can be implemented in agile development projects in order to minimize the impact of these commonly faced challenges. One of the good practices mentioned in the literature review is of pair programming. In pair programming two programmers or developers work together to design a software. One of the developers is known as the driver and the other is known as the navigator *(Begel et al., 2008)*. Both the developers work in collaboration to create the software product. However, the driver is responsible for writing the code and creating the designs. Whereas, the navigator is responsible for going through the work of the driver in order to identify defects or bugs *(Begel et al., 2008)*. Figure 13 summarizes the impact of pair programming on the three challenges of agile development discussed in this thesis.

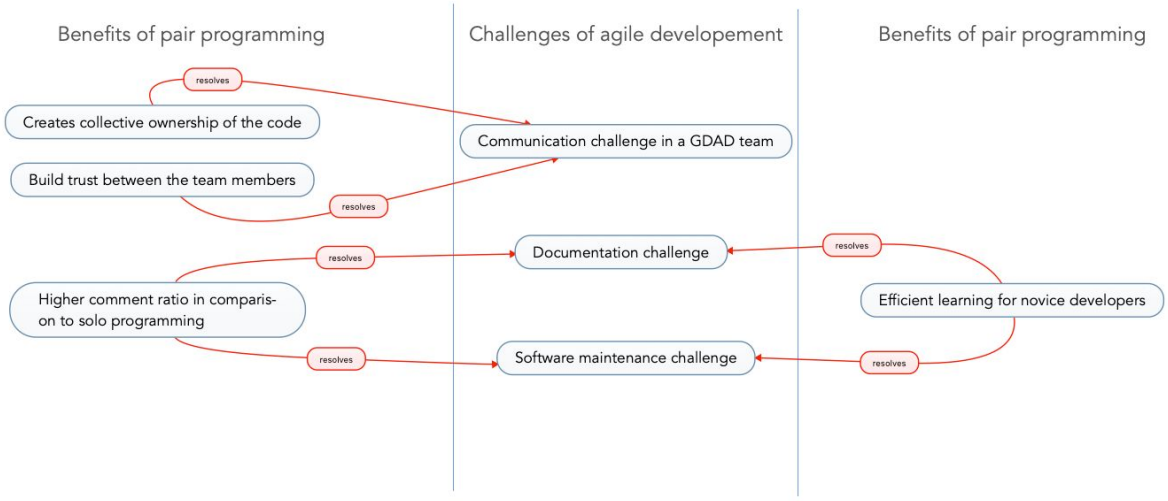

Figure 13. Impacts of pair programming

According to *Holmström et al, (2006)* pair programming helps in creating a collective ownership of the software being developed. Working in pairs makes it easier for the team members to understand each other and helps in creating a level of trust between them *(Dorairaj et al., 2011).* Trust helps in minimizing the communication challenges that stem from the distance difference in a geographically distributed team *(Holmström et al, 2006)*.

In agile development code is considered as the basis to understand the system behaviour as agile supports less documentation *(Fowler and Highsmith, 2001)*. Due to a lack of documentation new developers find it hard to understand the code behaviour especially when the code base is complex *(Hanssen et al., 2009)*. *Hulkko and Abrahamsson (2005)* concluded in their research that the code produced from pair programming has a higher ratio of code comments when compared to solo programming. Furthermore, *Begel et al. (2008)* discovered in their research that pair programming is good practice when the project team wants to onboard a new developer. It enables developers to learn new technologies faster and also provides an ability to learn from the developer they are working with *(Begel et al., 2008)*. Hence, pair programming can also be used to make the learning process of a new developer easier and efficient. Pair programming also results in a better software architecture design as there is another pair of eyes looking over the design to provide a broader insight *(Begel et al., 2008)*.

In summary the benefits of impleming pair programming in agile development are manifold. As per the finding of literature review implementing pair programming in agile development projects *caters to all the three challenges* that are discussed in this thesis. Firstly, pair programming creates trust between the team members and eventually reduces the communication gap within the team *(Holmström et al, 2006)*. Secondly, literature suggests that pair programming results in a higher code comment ratio and is also a more efficient way for novice developers to understand system behavior *(Hulkko and Abrahamsson, 2005)* . Lastly, it makes it easier to onboard new or novice developers in the team *(Begel et al., 2008)* and helps in creating a common understanding of the code within the team *(Holmström et al, 2006)*.

Another good practice discussed in this thesis is the creation of user stories in order to understand customer requirements. One core value of agile development is to focus on 'working software over documentation' *(Fowler and Highsmith, 2001)*. However, minimal documentation also results in some key challenges in agile development such as misunderstood customer requirements *(Cao and Ramesh, 2008)* and poor communication *(Inayat et al., 2015)*. Literature advocates to create user stories in agile development projects as a way to avoid heavy-weight documentation and yet have important customer requirements documented for future use *(Cao and Ramesh, 2008)*.

This thesis focuses on the template created by Cohn (2004) for user stories:

"As a *(type of user)*, I want *(goal)*, so that *(benefit or* why *is the change needed)*" *(Cohn, 2004)*.

The above template simply defines *who* is the user or who is it for, *what* does the user expect from the system and *why* is it important for the user *(Cohn, 2004)*. A user story tells the developers about the concrete requirements of the customer. The quality of the user stories is very important since developers use these stories to develop the product. There are a few frameworks in literature that aim to support developers in creating quality user stories. One such framework is known as INVEST *(Wake, 2003)* and is the most well-known framework *(Lucassen et. al., 2016)*. INVEST is an acronym that stands for the qualities a user story should have: Independent, Negotiable, Valuable, Estimable, Small and Testable *(Wake, 2003)*.

Lucassen et. al. (2016) suggests that user stories should be defined in collaboration with the customer. It is the responsibility of the customer to define the goal of the user story and explain *why* a particular change is needed *(Inayat et al., 2015)*. Hence, the customer should provide their input when the development team is creating user stories for the project *(Lucassen et. al., 2016)*. Once an initial draft of the user stories has been completed

the development team should seek the customers input regarding them. Based on the customer's feedback the user stories are refined in each iteration until all the customer requirements are clear for the development team *(Inayat et al., 2015)*.This constant interaction with the customer to understand their requirements helps in creating a shared understanding for both the customer and the development team *(Lucassen et. al., 2016)*. Eventually, having a shared understanding of the customer requirements results in a less number of defects and post-development changes requested by the customer *(Ramesh et. al., 2010).*

4. Empirical Study

The main aim of this section is to describe the findings of the empirical study in a manner to answer the first research question. The section is structured in a way to first explain the current organizational structure, describe the process followed for small-scale agile development projects and describe the customer success targets of the case company. This is followed with a view of the current challenges faced during small-scale agile development projects and their impact on customer sastisfaction. In the end, the chapter details the suggested actions for the case company to take in order to maximize customer sastisfaction.

4.1 Case company's organizational structure

The case company provides a supply chain management solution to its customers. The company mainly operates on a *Software as a Service delivery (SaaS)* model and has over 700 employees. The details regarding the solution will be not presented in this thesis as they are a trade secret. For the purpose of this thesis it is important to get an overall understanding of a customer's journey within the case company. Figure 14 shows an overview of the case company's functions and teams that lie in the scope of this thesis.

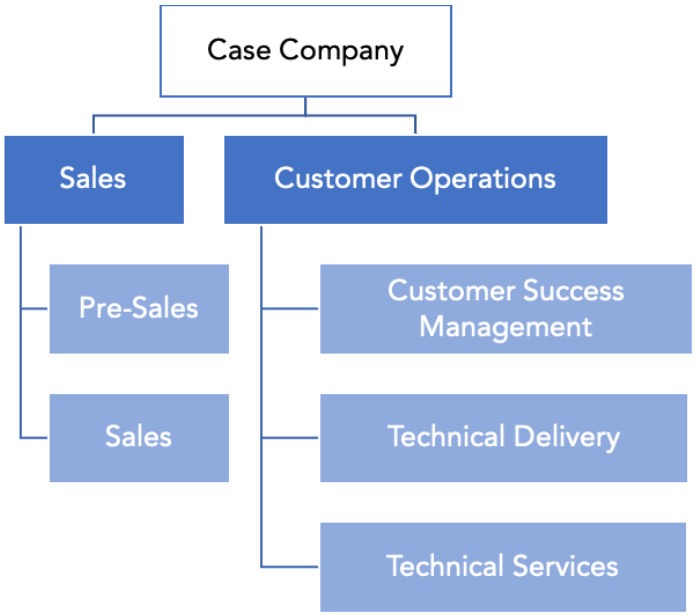

Figure 14. Functions and Teams at Case Company.

On an abstract level the company is divided into functions. There are two main functions within the company that are involved in the customer's journey namely, *Sales* and *Customer Operations*. Each function is then further divided into various teams. Table 11 gives a brief overview of each of the involved teams and their high-level responsibilities.

Table 11. Overview of teams and their responsibilities.

Team	**Responsibility**
Sales/Pre-sales	• Getting new customers • Present initial solutions to potential customers • Negotiate the terms of the contract with the customer
Customer Success Management	• Maintain customer satisfaction and a good relationship with existing customers • Bill the customer for small-scale agile development and other billable projects as per the contract
Technical Delivery Team	• Customize the solution to the customers' needs on the technical side and implement the technical integrations.
Technical Services Team	• Provide technical support during continuous service phase • Keep customer environments in good shape with reasonable efforts • Deliver good quality technical solutions to the customers in continuous service phases. These technical solutions are dealt as small-scale agile projects in the case company.

Sales function is responsible for pitching the case company's product to potential buyers and getting new customers. Sales function can be further divided into two teams Pre-Sales and Sales. Pre-sales is responsible for developing and presenting standard demos to the potential customers. Sales is responsible for negotiating and contracting with potential customers after the pre-sales team has shown them the product demos. Both these teams work closely together and can be considered as one team under the scope of this thesis.

Customer operations function is responsible for the service of the software product that is custom built and deployed for each customer individually. The software product that is custom built for each customer is called a *'customer environment'*. *Customer operations* further has separate teams to cater to varying customers needs such as business and technical needs. The responsibility to look after the general well-being of the customer and their relationship with the company falls under the duties of *Customer Success Management* team. The members of this team are known as *Customer Success Managers*. The technical well-being of the customer environment, such as implementation of customer specific requirements and version upgrades, is taken care of by the Technical *Deliver*y team and *Technical Services* team, depending on which specific phase the customer is in at the case company.

4.2 Customer journey phases at case company

In order to understand the process and the need for small-scale agile development in the case company it is first important to understand the various phases the customer goes through within the case company. The overall journey can be divided into three main phases:

1. Pre-sales/Sales phase
2. Project Delivery phase

3. Continuous service phase

Figure 15 gives an overview of the customer's journey within the case company.

Figure 15. Phases of customer's journey.

Figure 16 gives an overview of the teams that are involved during the different phases of the customer lifecycle.

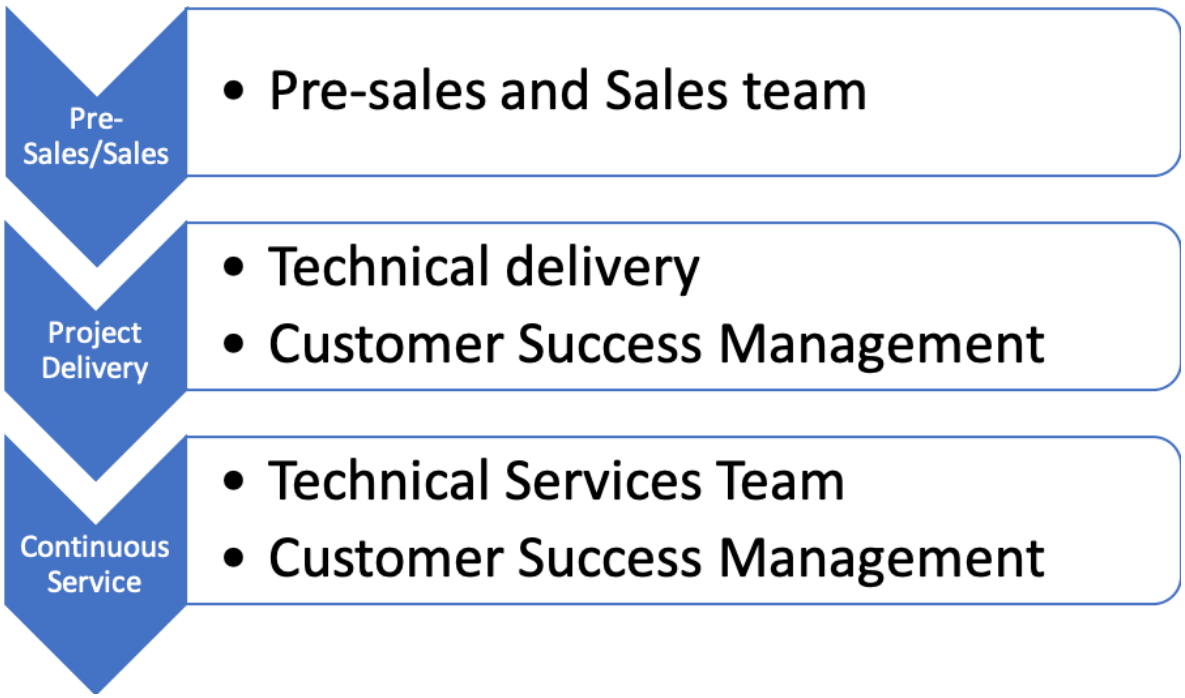

Figure 16. Phases of customer's journey and involved teams.

An individual customer's journey within the case company begins from the *Pre-sales/Sales phase* where the solution/product is pitched to potential customers and together with the customer the company tries to find a way to provide value. Once the customer is on board with the solution the Sales team negotiates the official terms of the contract with the customer (including the (Service Level Agreement) and signs the contract. After signing the contract, the customer now has a specified *Account Manager* within the company.

Once the customer has been signed the *Project Delivery phase* begins. During this phase the customer is assigned a dedicated *customer success manager* from the *Customer Success Management* team. Customer success managers are responsible for the general well-being of the customer. The main purpose of this phase is to set up the customer environment and implement the solution for the customer as per their business requirements. This is done by the *Technical Delivery* team. It can also be called the implementation phase and also involves the initial support and testing of the customer environment.

Towards the end of this phase, when a customer environment has been working in a stable manner for a while, the customer is moved to the *Continuous Service phase* and handed over to the *Technical Services* team. During the *continuous service phase* each customer has a dedicated technical and business team member looking after the individual needs of the customer and providing support based on the initially agreed SLAs. *Technical services team,* with the help of the customer success manager, provides consultancy and technical support to the customer as per the service-level agreements (SLAs). This thesis focuses on the *Continuous Service phase* of the customer's journey within the case company.

4.3 Continuous service phase in customer journey

As a part of continuous service phase the case company provides three main services:

1. Service Delivery,

2. Professional Services, and
3. Customer Success.

Service Delivery focuses on the contractual requirements for SaaS customers such as regular version upgrades of the software and non-billable support. *Professional Services* pertain to all the billable services that the company provides that are not a part of the SaaS contract such as implementing any change requests made by the customer. *Customer Success* focuses on the general well-being of the existing customers and ensuring that the company does not lose its customer base. An important part of customer success is to make sure that the solution provided by the company is maximizing value for the customer and to ensure outstanding customer satisfaction. Figure 17 explains the three services provided as a part of continuous service in more detail.

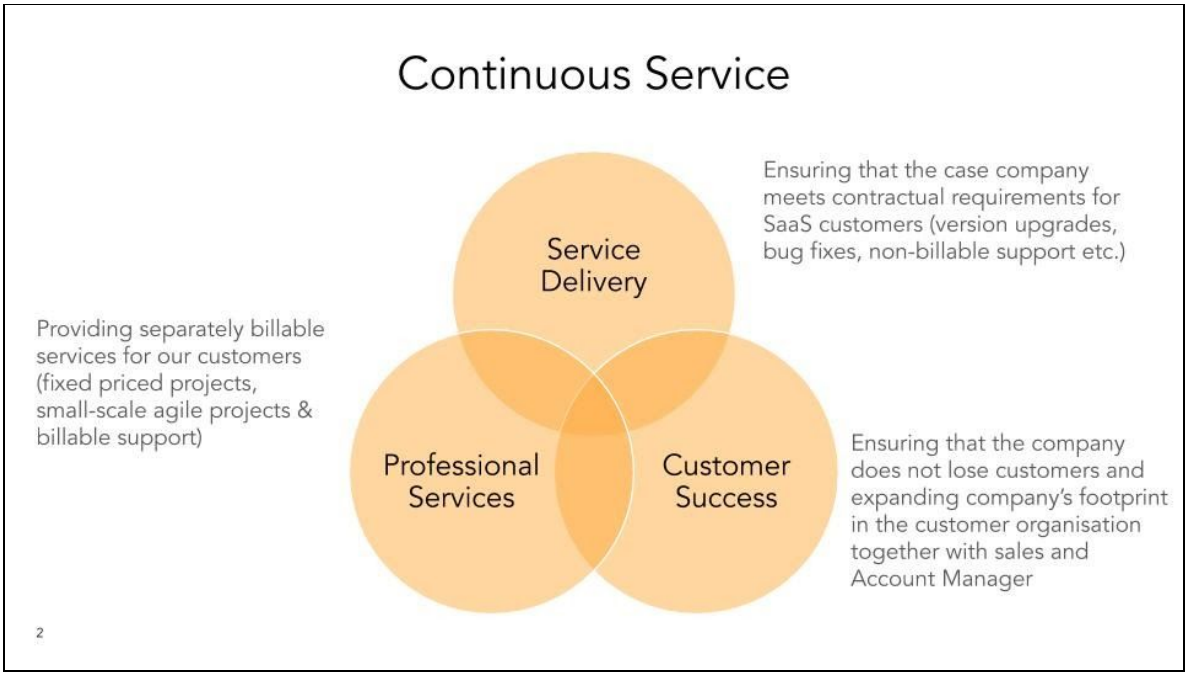

Figure 17. Services provided under continuous service phase .

4.3.1 Small-scale development projects as a part of continuous service

As a part of *Professional Services* the customer has the ability to request changes to the current product that they are using. *Change requests* made by the customer can be of technical or business nature. In order to scope this thesis it was decided to focus on technical change requests only.

Technical change requests involve making changes to the backend structure of the product in order to extend the current behavior of the product or to create new functionality within the product to cater the customer's changing needs. The need to have a technical change request can come directly from the customer or can also be initiated by the case company in order to maximize the customer's value. These technical change requests are dealt as ***small-scale agile development projects*** within the case company and are known as ***minor development projects***. The activities in such projects usually take less than 10 working days and the goal is to modify existing configuration to improve or introduce new capabilities within the customer environment. The end result of such projects has a considerable impact on the *customer loyalty* and *customer satisfaction* within the case company.

There are two main roles that are involved in such projects from the case company's perspective namely, *Customer Success Manager* and *Technical Consultant*. Table 12 gives a brief overview of the responsibilities associated with each role.

Table 12. Roles and responsibilities involved in small-scale agile projects at the case company

Role	Team	Responsibility
Customer Success Manager (CSM)	Customer Success Management	• Initiate, consult, plan and coordinate minor development work • Keep the customer happy and satisfied • Ensure a stable and well-functioning service with proactive support and transparent communication. • Take care of invoicing all billable tasks to the customer
Technical Consultant (TC)	Technical Services Team	• Deliver a working software project to the customer as per the agreed requirements • Take care of all software engineering activities required for the completion of the project • Provide a time estimate to the customer and the Customer Success Manager

Figure 18 defines the detailed process that is followed in order to complete these small-scale agile development projects (a.k.a. minor development projects).

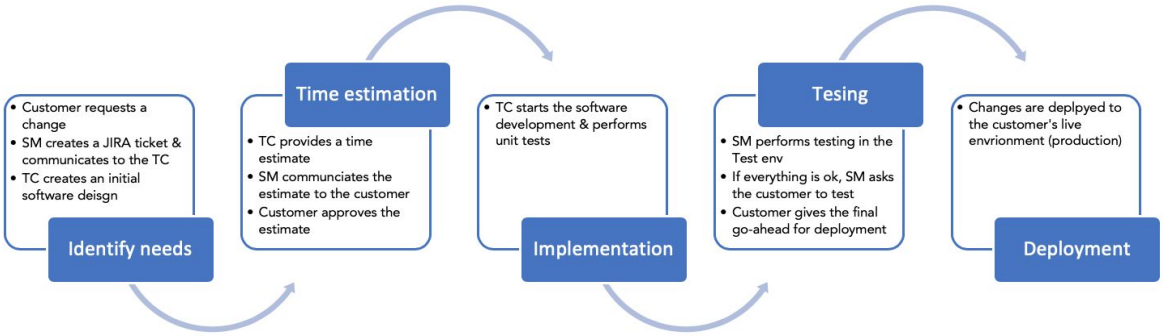

Figure 18. Small scale agile development process.

The first step is to identify the need behind the change request. During this phase the *customer success manager (CSM)* works closely with the customer to understand their business needs. After this the CSM creates a Jira ticket to document the customer requirements and communicate them to the *Technical Consultant (TC)*. The TC then goes through the documentation and creates an initial software design based on the specified requirements. Based on this design, the TC provides a time estimate that is required to finish the project to the CSM. The CSM then communicates this estimate to the customer. Based on this estimate the customer is eventually billed according to the amount that is specified in each customer's contract.

Once the customer accepts the time estimate, the development of the project is started. The TC then implements the solution locally and performs the desired tests. The changes are then deployed in a test environment where the initial acceptance testing is performed by the CSM. The customer vigorously performs user testing on the test environment once the CSM has tested and approved the changes. As soon as the customer approves all the changes, the project is then deployed in the live customer environment (also called the production environment) and the project is completed.

4.3.2 Customer satisfaction as a part of continuous service

An important part of the main research problem is to focus on the impact small-scale agile projects have on customer satisfaction. In this case company customer satisfaction comes under the umbrella of customer success. For this reason, it is important to understand what customer success means in the case company and the customer success targets is the case company trying to achieve. Case company has defined a few targets that help them in achieving customer success. Figure 19 gives an overview of the customer success targets at the case company.

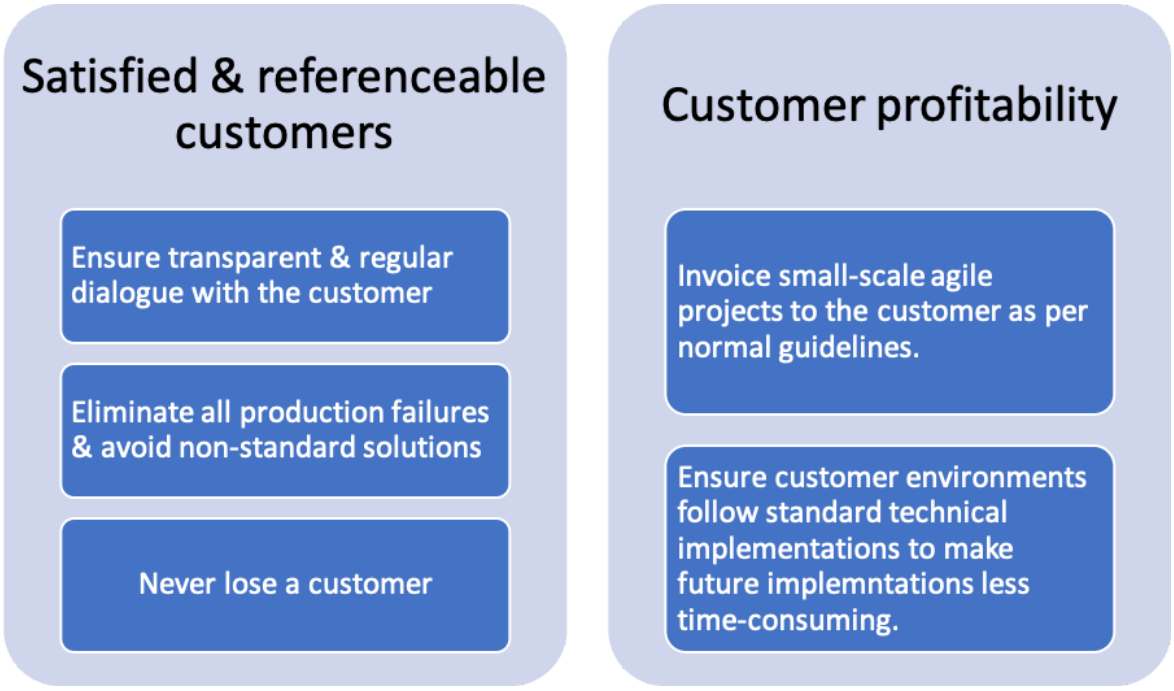

Figure 19. Customer success targets of case company

From the perspective of case company customer success can be defined as the set of practices that aim at ensuring that the company does not lose customers and expands its business with customers. These practices also aim at making sure that customer referenceability is at its maximum while keeping the customer profitable as well. Customer referenceability means that the case company's current customers are comfortable in sharing positive experiences about how the company's product has helped their business

grow. Furthermore, an important aspect of customer success is to ensure outstanding customer satisfaction by maximizing customer's business value

4.4 Current state of small-scale agile development projects

4.4.1 Challenges of small-scale agile development project

This section is based on the interviews that were conducted as a part of the empirical study. The main aim is to answer the first research question by exploring the challenges of small scale agile projects at the case company and their impact on customer success. For this reason, in this section we summarize the common challenges faced by the case company employees during small-scale agile development projects. Both the involved roles, *customer success manager and technical consultant*, were given a fair representation in the interviews. The raw data gathered from the interviews will not be shared publicly as it contains confidential information regarding the case company.

The raw data was first organized into three main categories namely *challenges, impact on customer success and suggestions.* After that the data was further filtered to identify five major challenges that the case company faces in small-scale agile development projects. These set of major challenges are further described in Table 13.

Table 13. Challenges of small-scale projects at case company

Challenge	Description
Legacy implementations	A few customer environments still have legacy implementation. Legacy implementations can prolong the development process due to lack of knowledge regarding such implementations.
Poor load balancing	Technical consultants are not well-equipped to balance their work load. A TC is responsible for more than 1 customer. Working on simultaneous projects for different customers can lead to mistakes and unmet deadlines.
Delayed project delivery	TCs are sometimes not able to provide accurate work estimates for projects due to several reasons such as legacy implementation and lack of product and customer knowledge. Inaccurate time estimates can make projects exceed their deadlines which results in customer dissatisfaction.
Poor communication	Due to distance differences it is difficult to arrange face-to-face meetings within the project teams. Lack of face-to-face communication within the project team can create misunderstandings
Undefined process	There are no standard development guidelines defined for small-scale agile projects. It practically means that each TC works on projects as they see fit i.e. create documentation on their own format, comment the code as they see fit etc. Lack of standard guidelines for the development process, such as coding practices and documentation, can lead to problems.

One of the major issues identified from the empirical study was of legacy implementations. The product is constantly evolving in order to cater to the changing customer needs and technical trends. Due to this, there are quite a few legacy implementations present in some older customer environments such as data transfer over old protocols. Such legacy implementations are no longer applicable to the current product implementation and have become obsolete. In such cases doing a small change to the customer's environment can take a lot of time due to lack of knowledge around these legacy implementations. The team members who have knowledge regarding such legacy implementations might not have the time to share their expertise or in some cases might have even left the company.

Another problem identified during data analysis was poor load balancing at the part of technical consultants. A single technical consultant is responsible for more than one customer. Customer requests come in at an unpredictable rate during the *continuous service phase* which makes it difficult to predict the future workload of a technical consultant. Since TCs are responsible for multiple customers this means that they are at times working on multiple projects simultaneously as well.The chances of careless mistakes by the technical consultants are increased due to different ongoing projects at the same time. This issue of load balancing sometimes even leads to pushing forward some projects and disrupting the initial schedule for them.

An important problem identified during data analysis was that technical consultants (TC) sometimes fail at providing a correct time estimate at the beginning of the project. There are several factors that contribute to this. Few of the important factors that were discussed in the interviews are:

- It can be difficult to provide an accurate time estimate due to lack of customer-specific and product-specific knowledge especially for employees with less experience.
- Some customer environments have outdated setups or legacy implementations. Lack of knowledge about customer-specific legacy implementations can result in inaccurate time estimates.

- Lastly, TCs often do not explicitly discuss the expected deadline with the *customer success manager or the customer*. This leads to misunderstood expectations about the deadline.

Poor communication within the project team is another problem that was identified during data analysis. Case company has offices in around 10 counties. This practically means that the team working on the project is mostly distributed globally. Due to this internal communication is heavily done on company provided VoIP mediums. This makes the overall communication process inefficient and complicated. Furthermore, customer success managers behave as the main communication link between the customer and the TC. This lack of direct link between the TC and the customer further adds to the inefficiency of the communication and results in loss of meaning.

Lastly, an interesting challenge that came forward during interviews was the lack of standard guidelines regarding the development process of small-scale agile development projects. Recently, a well-thought process was released for change management (including small-scale agile projects) and is based on JIRA. However, the current JIRA process does not define a standard set of guidelines for design documentation, coding standards, and peer-review practices. There are no standard guidelines for documenting the changes done as a part of small-scale agile projects. This in practice means that the level of documentation done for each project is different and depends entirely on the team members. Technical consultants use different ways to comment and format the code which results in a poor documentation of the code. Furthermore, lack of standard peer-review guidelines can often result in the deployment of faulty changes to the customer environment. Lack of peer-review and proper testing can often result in the deployment of faulty changes to the customer environment. This can delay the delivery of the project if such risks were not accounted for in the initial time estimate.

4.4.2 Challenges of small-scale agile development project & customer satisfaction

All the identified challenges are interlinked and have a negative impact on customer success within the case company. Both *legacy implementations* and *load-balancing* result in delayed delivery of small-scale agile projects. For this reason we decide to group both these challenges under *delayed project delivery* when evaluating the impact of these challenges on customer success. Figure 20 gives an overview of the factors that have a negative impact on customer success.

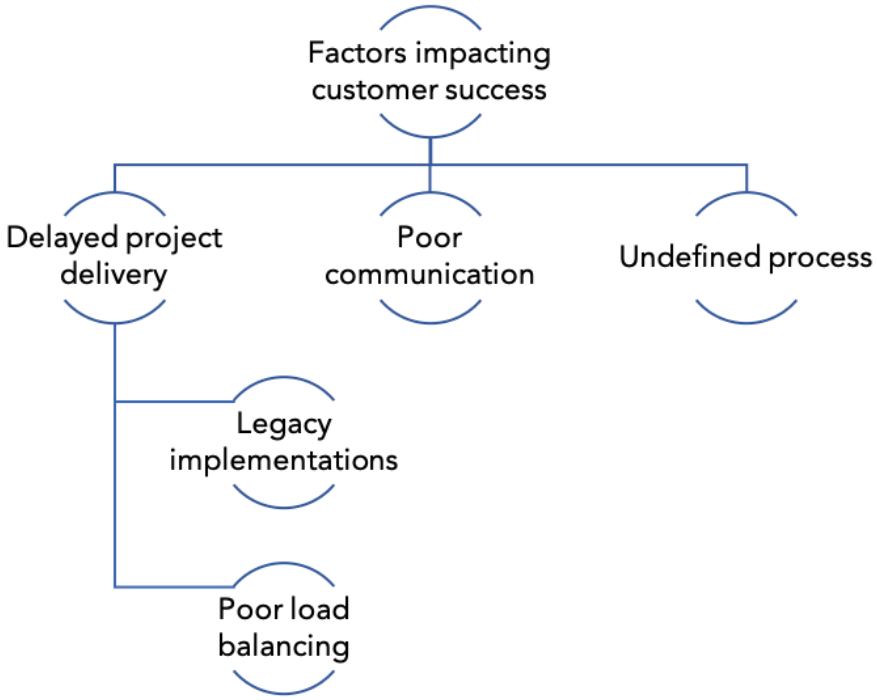

Figure 20. Challenges impacting customer satisfaction at case company

Delayed delivery of small-scale agile development projects seems to be the factor that impacts customer success the most. The main reason behind delayed project delivery, as uncovered in the interviews, is lack of customer and product-specific knowledge. The product is evolving with time in order to cater to the changing customer needs and technical

trends. In practical terms this means there are some old customer projects that have outdated integrations or legacy implementation in place. Examples of such implementations can be direct database integration and data transfer over older protocols. The documentation on such legacy implementations is hard to find. In some scenarios, the team members who implemented the project might have left the company. In such customer environments it can be hard for the TCs to estimate an accurate time for small-scale projects. Furthermore, it can be hard for the technical consultants to predict their workload due to poor load balancing within the case company. This in practice can mean that an ongoing project might get delayed due to some priority customer request that came up unexpectedly. This has a direct negative impact on customer satisfaction and profitability.

Poor communication has a negative impact on customer success and there are several factors that contribute to poor communication. There is little or no face-to-face communication between the customer success manager and technical consultant as the project team is globally distributed. Communication over VoIP or other online mediums can often result in misunderstood requirements. Furthermore, in most cases customer success managers act as a communication link between the customer and the technical consultant. This further adds to the problem of misunderstood requirements. Customer success managers might not be able to communicate the exact customer need to technical consultants. All of the above factors combined often result in technical consultants implementing solutions that do not provide any real value to the customer. One of the key customer success targets for the case company is *delivering measurable value* to the customer.

The challenge of *undefined process* revolves around the absence of standard guidelines regarding design documentation, code documentation, and peer-review practices. Before starting a small-scale development project, TCs need to understand the code logic implemented within a customer environment. If the existing code logic for the customer environment is non-standard and poorly documented it can be time consuming for TC to proceed with the project. This in turn can also result in late deliveries of projects and

inaccurate time estimates. Lack of peer-review results in deploying a faulty or inefficient solution on the customer's environment. Issues that come up in a customer's environment due to faulty deployment can result in monetary loss for the customer. In the longer run this can have a negative impact on the customer's relationship with the company and their willingness to reference case company products to other businesses.

Figure 21 maps the above challenges to the customer success targets of the case company.

Figure 21. Mapping of challenges to customer satisfaction targets of case company

Table 14 summarizes the impact each identified challenge has on customer success within the case company.

Table 14. Impact of identified challenges on customer satisfaction at case company

Challenge	**Impact on customer satisfaction**
Delayed project delivery	Inaccurate time estimates can lead to late delivery of projects which results in customer dissatisfaction. *Legacy implementation* and *poor load balancing* are underlying factors for late project deliveries. This results in customer dissatisfaction and lowers customer profitability.
Poor communication	Lack of direct communication between the customer and the technical consultant leads to misunderstood requirements. This results in implementing solutions that do not provide value to the customer. This in turn results in customer dissatisfaction.
Undefined process	Lack of peer-review and proper testing can often result in the deployment of faulty changes to the customer environment. Such small careless mistakes can sometimes result in monetary loss for the customer especially when they occur in a production environment. Such incidents can lead to customer dissatisfaction which can impact the customer's relationship with the case company. In some cases it can also result in the company losing customers.

4.5 Suggested good practices

4.5.1 Good practices to minimize delayed project delivery

Figure 22 gives a brief overview of the good practices suggested by this thesis to reduce delayed project delivery

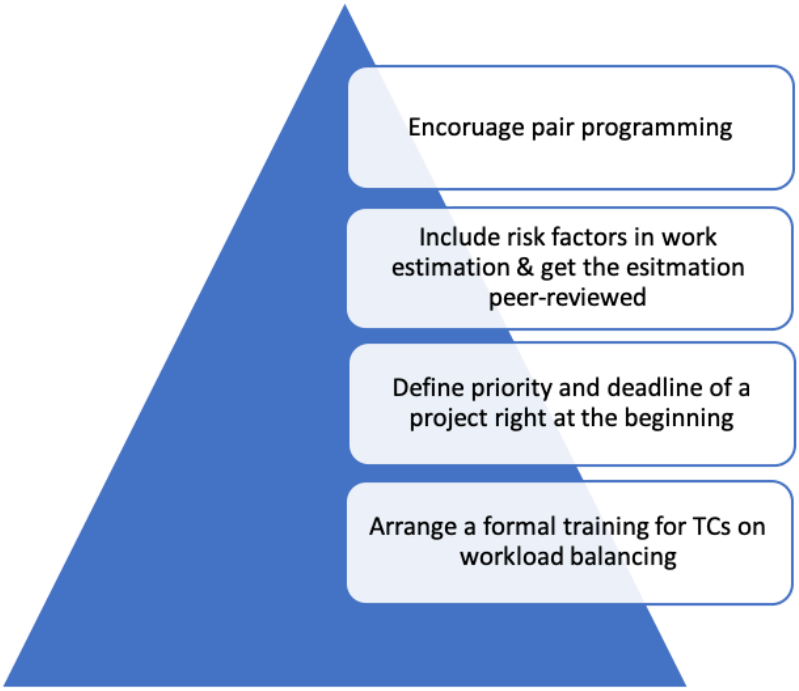

Figure 22. Overview of good practices to reduce delayed project delivery

Small-scale agile development projects (or minor development projects) in the case company require product-specific and customer-specific knowledge. Lack of such knowledge makes it difficult for new or junior technical consultants to work on projects. Pair programming should be practiced in order to deepen the understanding for less experienced or new technical consultants (*Deursen, 2001*). *Hulkko and Abrahamsson (2005)* suggested in their research that pair programming increases the confidence of the developers involved. *Lui and Chan (2008)* suggested that developers work more efficiently in pairs when faced with a complex or difficult problem to solve. Hence, this thesis

encourages the pairing of two technical consultants especially when working on complex projects. The pairing can be done in multiple formats such as a more experienced technical consultant can be paired with a junior technical consultant. An added benefit of this approach is that it will help in creating a collective ownership of the project *(Holmström et. al, 2006)*. Furthermore, it will also help the new team members in getting a deeper understanding of the product they are working on *(Begel et al., 2008)*.

A delayed project delivery has a negative impact on the relationship between the customer and the company. Data analysis of the empirical study points out that not having a common understanding of the priority and deadline of the project often leads to load balancing issues for technical consultants. Such load balancing issues further results in a delayed project delivery at times as well. It was suggested in the interviews that technical consultants should ask the customer success managers to provide the expected deadline and priority of the project towards the beginning. The customer success managers should always define the priority after consulting the customer. The small-scale agile projects conducted in the case company are done for individual customers. Hence, the priority of the project should be defined by the customer for which the project is being conducted. This can help in avoiding misunderstandings regarding the prioritization and can also make it easier for the technical consultant to balance their workload.

Another way to avoid the delay in project delivery is to make sure that all risk factors are accounted for when work estimations are being created by the technical consultant. This suggestion was raised during the interviews for this thesis and is also pointed out by *Hanssen et al. (2009)* in their work. Technical consultants should be encouraged to keep in mind the complexity of the task while creating work estimations and plan accordingly. However, new or less experienced technical consultants might not always be able to consider all the aspects of the project thoroughly. Hence, it was suggested in the interviews that a senior technical consultant should peer review the work estimations provided by junior team members for a project. This would ensure that all possible risks are accounted

for in the work estimation and would also be a good learning ground for the new team members.

Lastly, it was suggested in the interviews that a formal training should be conducted for the technical consultants to help them understand how they can better manage their tasks. One suggestion that came forward in the interviews was that technical consultants should delegate their tasks if they feel they cannot meet the deadline. Furthermore, based on data analysis done in empirical study, this thesis suggests that technical consultants can create a dashboard in JIRA that displays the tickets that have been neglected for a given period of time. This will give the technical consultant more visibility of the pending projects or tasks and help them balance their workload.

4.5.2 Good practices to minimize poor communication

Figure 23 gives a brief overview of the good practices suggested by this thesis to reduce poor communication.

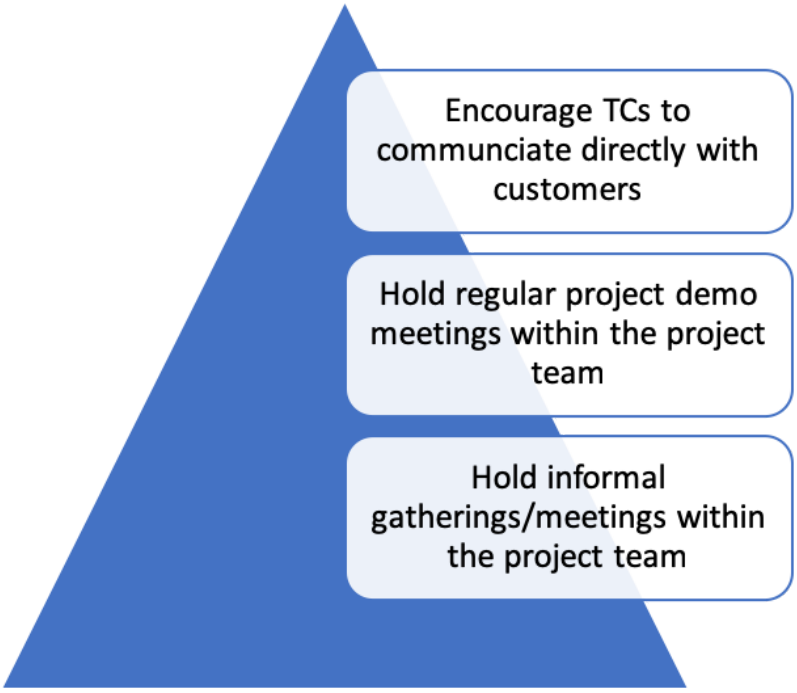

Figure 23. Overview of good practices to reduce poor communication

The data analysis from the empirical study points out that customer success manager (CSM) acts as a buffer between the customer and the technical consultant. This thesis suggests that direct interaction between the customer and the technical consultant should be encouraged especially towards the beginning of the project when the requirements are being defined. This suggestion stems from the literature and was also brought forward during the interviews. *Cao and Ramesh (2008)* specifically point out that the quality of the requirement specifications depend on the quality of interaction between the developers and the customer. Hence, technical consultants should be put in direct contact with the customer when defining the requirements of the project. Online communication tools should be used in case it is difficult to arrange face-to-face meetings.

The project team at the case company is geographically distributed i.e. the CSM and technical consultant are usually not located in the same geographic location. Hence, this thesis encourages regular project demonstration meetings between the project team in order to make sure that everyone has the same understanding of the task at hand *(Green et. al., 2010)*. This suggestion was also brought forward in the interviews. Regular online meetings (or face-to-face when possible) should be arranged within the project team to make sure that everyone in the team has up-to-date knowledge about the project *(Sindhgatta et. al., 2011)*. These meetings come with an additional benefit of making team members comfortable with each other as they can read each other's body language *(Dorairaj et al., 2011)*. Furthermore, online meetings should also be recorded so that absent team members can get up-to-date.

Lastly, this thesis suggests the case company to encourage informal meet-ups or online meetings between the two roles that are actively involved in a small-scale agile development project i.e. customer success manager and the technical consultant. This suggestion stems from *Dorairaj et al. (2011)* work where they pointed out that informal conversations form the basis of strong work relationships. Hence, the case company

should encourage and establish informal conversations within the project team members by arranging online meetings or face-to-face meetups when possible. Currently, such meetings are frequently arranged between the technical consultants as it was pointed out in the interviews. However, we feel it is important to hold such informal gatherings between the customer success manager and the technical consultant working on the same project in order to create a good working atmosphere.

4.5.3 Good practices to defined a process

Figure 24 gives a brief overview of the good practices suggested by this thesis to address the challenge of undefined software development process.

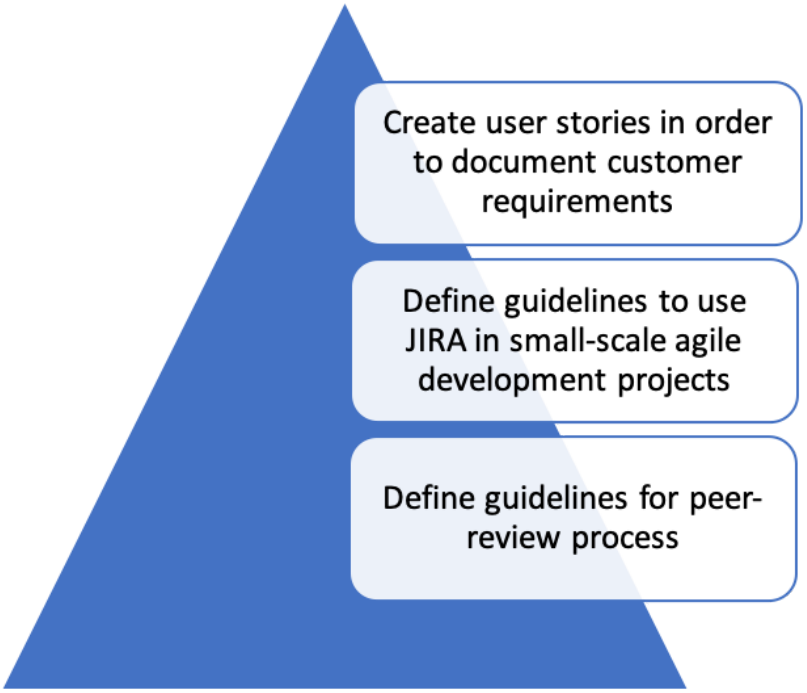

Figure 24. Overview of the good practices create a process

It was suggested in the interviews that at the beginning of the project the customer success manager and the technical consultant should clearly define their responsibilities. This thesis also suggests that concise documentation of the changes should be created before starting the development process. *Lucassen et. al. (2016),* suggested that creating concise user

stories help in creating a shared understanding of the requirements. Hence, this thesis encourages the creation of concise documentation of customer requirements at the beginning of the project. The technical consultant and the customer success manager should work on creating a concise documentation along with the customer in order to create a shared understanding of the project. This documentation can be done in the form of user stories. The template on which the user stories can be documented should be created by the team itself as the template itself does not matter as long as the whole team is using the same template to document user stories *(Lucassen et. al., 2016)*. This thesis suggests the template created by Cohn (2004) for user stories:

"As a *(type of user)*, I want *(goal)*, so that *(benefit or* why *is the change needed)*" *(Cohn, 2004)*.

The documentation of user stories can be done on the JIRA ticket itself or can be done separately but then should be attached with the JIRA ticket. This will help in keeping track of the requirements and the progress of the project. A user story should be atomic and unique as defined by *Lucassen et al. (2015)*. Each user story should have an estimate associated with it as well. *Wake (2003)* suggests that user stories should be estimable, small and testable. This thesis suggests that technical consultants should document the schedule and work estimation of each user story as additional notes attached to it. In JIRA the information about the schedule and work estimation can be added to the specific fields defined for this purpose.

Layman et. al. (2010) recommended for GDAD (geographically distributed agile development) teams to use globally-available project management tools to record and monitor the current status of the project. The case company recently started using JIRA for this purpose. However, the data analysis done for this thesis suggests that JIRA is still not fully utilized when it comes to small-scale agile development projects. It was suggested in the interviews that JIRA should be further explored in order to fully utilize the tool and

balance the workload of a technical consultant. Based on this suggestion a new thesis research topic was initiated at the case company during the course of this project.

Lastly, this thesis suggests that a formal process for peer reviews should be created based on the suggestions that came forward during the interviews. A few aspects that can be considered while creating the guidelines can be how can a TC select a reviewer, when can a TC bypass the peer-review process etc. It would make it easier to ensure that everyone in the team is following the defined standards and have the same understanding of the common standards. It should also be encouraged to pick or assign a reviewer right towards the beginning of the project. Furthermore, this thesis also suggests that the case company should create a checklist or set of standard guidelines for the reviewers as well. These set of standard guidelines should be created in order to facilitate the reviewer in the process of peer-review and make sure that they do not skip one of the basic quality checks while doing the review.

4.5.4 Summary of suggested good practices

The suggested good practices presented in this chapter are intended to help the case company maximize customer satisfaction while working on small-scale agile projects (termed as minor development projects within the company). The suggested good practices are according to the three main challenges that were identified in the empirical analysis: Delayed project delivery, poor communication and undefined process. The suggestions made as part of this thesis are based on the literature review. However, a few suggestions also stem from the suggestions that were discussed during the interview. The thesis clearly defines the source of the suggested good practices in the upcoming sections.

The main aim behind the suggested good practices is to minimize, or in some cases to eliminate, the challenges that were identified as part of this thesis. Table 15 has the suggested good practices in a concise format.

Table 15. Suggested good practices to minimize identified challenges at case company

Challenge	Good practices to maximize customer satisfaction
Delayed project delivery	● Encourage pair programming in complex projects ● Encourage technical consultants to account for risks associated with a task or project while creating work estimations. Furthermore, technical consultants should get their work estimations peer reviewed. ● Encourage the project team to define priority and deadline of the project at the beginning. ● Arrange a training for the technical consultants to help them understand work load-balancing.
Poor communication	● Encourage technical consultants to directly communicate with the customer ● Encourage the project team to hold regular project demonstration meetings. ● Arrange informal meet-ups or online meetings between the project team on a regular basis.
Undefined process	● Encourage the creation of concise documentation of customer requirements in the form of user stories. ● Define guidelines to use JIRA in small scale agile development projects (minor development projects). ● Encourage the project team to clearly define the roles and responsibilities at the beginning of the project. ● Define guidelines for peer review process.

5. Discussion

This chapter discusses the answers to the two research questions that were defined towards the beginning.

5.1 RQ1: Current state of small-scale agile development at the case company

The first research question of this Thesis is *What is the current state of small-scale agile development at the Case Company*? In order to answer this question we first aimed at creating an understanding of the process that is currently being followed for small-scale agile development projects at the case company. Alongside this we also identified the challenges that these projects faced and how these challenges impacted customer satisfaction. As a result of the analysis, we were able to identify 3 unique challenges that small-scale agile development projects faced at the case company.

The identified core challenges are:

1. The delivery of small-scale agile development projects is often delayed in the case company due to legacy implementation, poor workload balancing and lack of customer or product knowledge.
2. Customer requirements and expectations are often misunderstood due to lack of communication between the technical consultants and the customer.
3. The overall process for small-scale agile development does not define any guidelines that technical consultants can follow in regards to documentation, code commenting and peer-reviews.

One of the core reasons identified behind delayed project deliveries was legacy implementations that are still used by some customer environments. As *Cao and Ramesh (2008*) pointed out, over time the software architecture evolves and the implementation that was done towards the beginning of the project becomes legacy. It can get difficult to

understand code behavior of legacy implementation as the code base starts to get big *(Cao and Ramesh, 2008)*. A similar problem was seen in the case company as well. There was little or no documentation regarding the legacy implementations. In most cases the team members who worked on the legacy implementation have left the company or do not have the time to help with the project. This lack of documentation regarding legacy implementations makes it cumbersome to work on such customer environments and adds time to the overall development process.

Another root cause of delayed project delivery was that technical consultants (TC) sometimes fail at providing a correct time estimate at the beginning of the project. Delays in project delivery can happen due to a number of reasons such as lack of customer or product specific knowledge and legacy implementations. *Hanssen et al. (2009)*, also pointed out that at times making small changes to legacy implementations can take a lot of time and hence can slow down the overall development process. Complex or legacy code can also make it difficult for a new developer to understand the code behaviour and in most cases would require a senior developer's help (*Hanssen et al., 2009*). Furthermore, another factor contributing to the delayed delivery of projects is that technical consultants are responsible for more than 1 customer environment at the same time. The workload coming from customers is unpredictable and a technical consultant is working on different projects simultaneously. This makes it difficult to estimate workload of a technical consultant and eventually leads to poor workload balancing issues. Poor workload-balancing can lead to low quality of work and delayed projects.

Moreover, technical consultants do not explicitly discuss the deadline of the project with the customer success manager. This poor communication regarding the deadline with the customer success manager leads to a difference in the expectations of the customer/customer success manager and the technical consultant. In a longer run these challenges result in customer dissatisfaction and over the time lowers the customer profitability for the case company.

Another major challenge that small-scale agile development faces at the case company is poor communication. Customer success manager acts as the contact point for both the customer and technical consultants. Case company has offices in around 10 countries which implies that the project team itself is not always working from the same geographic location. Ibrahim et al. (2016), pointed out that it can get difficult to communicate requirements with customers that are in a different time-zone or geographical location. The analysis of the empirical data collected as a part of this thesis also supports this observation made by Ibrahim et al. (2016). The communication between customer success managers and technical consultants is not face-to-face as they usually work in different geographic locations. This lack of communication often leads to misunderstanding regarding different aspects of a project such as customer requirements and deadline. Delays in project deliveries and wrong implementations creates an overall bad experience for the customer.

Lastly, there is a lack of guidelines regarding the small-scale agile development projects. Even though recently the case company created a well-thought process for change management, including small-scale agile projects, it still does not lay out guidelines that can be used for documentation, code commenting and peer-reviews. Documentation for each project is done in different formats and the details introduced in the documentation depends on each individual that is working on the project. A side effect of this poor documentation of projects is that it becomes difficult for a junior or new technical consultant to understand the customer environment. Eventually this becomes the underlying cause for the legacy implementation challenge that we discussed above. Furthermore, there is no standard procedure defined for peer-review which eventually leads to the deployment of faulty changes to the customer environments. Issues resulting from faulty deployments can often result in monetary loss for the customer and impact the relationship of the company with the customer in the longer run. In some cases the case company might even lose customers due to these issues.

5.2 RQ2: Good practices to maximize customer satisfaction

The second research question was *What good practices can be used by the case company in development projects to maximize customer satisfaction?* The main aim of this question was to list good practices that the case company can implement in order to maximize customer satisfaction. This thesis suggested good practices related to the 3 main challenges that were identified as a result of the first research question.

5.2.1 Good practices to minimize delayed project delivery

One of the most important good practices suggested as a part of thesis is to encourage pair programming especially in the case of complex projects. Pair programming can be used to eliminate multiple challenges such as inaccurate time estimates, lack of product or customer specific knowledge and lack of peer-reviews. Another benefit of encouraging pair programming is that it can help software developers deepen their understanding regarding particular aspects of the product *(Deursen, 2001)* and can also help gain customer knowledge. In nutshell, by encouraging pair programming the case company can reduce the risk of mistakes as each aspect of the project would be reviewed by two pairs of eyes.

Secondly, technical consultants should keep in mind the risks associated with a project while creating work estimations as suggested by *Hanssen et al. (2009)*. However, it can be daunting for junior technical consultants to provide time estimates for projects especially since they lack customer and product knowledge. This thesis suggests encouraging junior technical consultants to work in pair with a senior or more experienced technical consultant while creating estimations. If that is not possible then junior technical consultants should be encouraged to get their time estimates peer-reviewed from senior team members. This practice would help in avoiding unrealistic estimates and make sure risks are considered while calculating the estimates.

Another suggestion to help with delayed delivery of projects is to encourage technical consultants to communicate the priority and deadline of a project with the customer success manager at the beginning of the project. *Inayat et al. (2015)* however points out that priority of a task should be defined by the customer. Hence, this thesis suggests that the customer should define the priority of the project and this should then be communicated to the technical consultants as well. This can eliminate misunderstandings regarding the deadlines and can also help technical consultants in balancing their work more effectively. This suggestion was pointed out during the interviews that were conducted as a part of the empirical study.

Lastly, this thesis suggests that formal training should be arranged for technical consultants to help them understand how they can effectively manage their workload. A set of standard guidelines should be created that can eventually help technical consultants in prioritizing multiple projects simultaneously. This suggestion was pointed out in the interviews that were conducted as part of the empirical study.

5.2.2 Good practices to minimize poor communication

The first suggestion to help with poor communication is to encourage technical consultants to communicate with the customer directly or to make sure that technical consultants are at least a part of the requirement specification meeting with the customer. This can help in making sure that everyone in the project team understands the customer's requirements and are on the same page. *Cao and Ramesh (2008)*, also pointed out that the quality of requirements specification is in direct relation with the quality of interaction between the customer and the developers. However, in the case of the case company it might not always be possible for technical consultants to meet the customers due to different geographic locations and timezone. Hence, this thesis suggests that online communication methods should be encouraged in case the technical consultant is from another geographic location.

Furthermore, it should be encouraged to hold regular project demonstration meetings especially when the project team is located in separate geographic locations. This would ensure that the whole project team is on the same page in regards to the project and has updated knowledge as well. The online meetings should be recorded so that absent team members can listen to the recording in order to get up to speed with the progress of the project. This suggestion is based on the findings from the works of *Green et. al. (2010), Sindhgatta et. al., (2011) and Dorairaj et al., (2011).*

Lastly, this thesis suggests to encourage informal meetings between the project team i.e. the customer success manager and the technical consultant. *Dorairaj et al., (2011),* pointed out that having informal meetings is very important to create strong working relationships within a team. The purpose of this is to make the team members comfortable with each other and establish a stronger working relationship between the project team. Informal meetings can also be arranged by using online communication tools and it is not necessary to rely on face-to-face informal meetings.

5.2.3 Good practices to define a process

One of the most important suggestions made in this thesis is that concise documentation of the project should be encouraged and guidelines should be created to document the changes. This thesis suggests to utilize creation of user stories in order to accomplish concise documentation of the project as also suggested by *Lucassen et. al. (2016)*. User stories should be created in collaboration with the customer and the customer should be asked to define a priority of each user story. This practice can help the project team and the customer to have a shared understanding of the customer requirements. Furthermore, there should be standard guidelines defined to help the project team while doing the documentation for example a standard format can be created that can be used by everyone throughout the case company. This will ensure that the project team caters all the customer

requirements and it will also make it easier for new team members to understand the project.

Furthermore, the case company recently started using JIRA, a globally available project management tool, to record and monitor the progress of their projects. One of the most important suggestions made in this thesis is to create a well-defined and standard process for using JIRA as a way to document and monitor projects. This suggestion came forward during the interviews as well. A well-defined JIRA process can in longer term help technical consultants to balance their workload as well. Everything regarding the project should be documented on the JIRA ticket such as the deadline, priority, work estimation and documentation. The project team should be encouraged towards the beginning of the project to clearly define their roles and responsibilities mutually. This information should also be included in the JIRA project ticket.

Lastly, and importantly, a suggestion came forward during the interviews that the case company should work on creating peer-review guidelines and technical consultants should be encouraged to get their work peer reviewed. We feel that creating standard guidelines on how a peer-review process should work will make it easier for the technical consultants to implement peer reviews in their projects. There should also be a set of standard guidelines defined for the person performing in the peer-reviews. This would create multiple benefits for the peer-reviewer such as ensuring that they do not miss out an important aspect while reviewing the project and make the whole process more effective.

5.3 Limitations

The good practices suggested in this thesis have not been validated in the case company due to limited available time. Implementing the results of the thesis in practice and observing their impact would have made it easier to validate the quality of these suggestions. Hence, it is hard to say if the presented practices will prove to be effective for the case company or not.

This thesis was completed over a period of nine months. The long duration of the thesis poses a risk that some of the factors that helped us in deciding the scope of this thesis towards the beginning might have evolved over the period of time. One good example of this can be the sudden change in the work environment that resulted from the global pandemic Covid-19. In the middle of this thesis the world got hit by a global pandemic known as Covid-19. Due to this pandemic the work practices at the case company saw a major shift as everyone in the company started working remotely from their homes. As the interviews had already been conducted by the time Covid-19 emerged, the impact of this change was not incorporated in the results of this thesis. However, this thesis does take into account a work environment where teams are working from different geographical locations and communicate via online communication tools. This gives us the confidence that the reasoning used behind suggested good practices still remains valid. But as the results have not been validated in a practical setting it is hard to say if the good practices suggested by the thesis can be utilized effectively or not.

6. Conclusions

The main aim of this thesis was to do a comprehensive study on how SaaS companies can maximize customer satisfaction from small-scale agile development projects. The empirical study for this thesis has been conducted in a Finnish SaaS company that provides a supply chain management software product to her customers. The research approach taken to conduct this study was action research. This chapter first focuses on the key conclusions that can be drawn from this study and then discusses the future research that can be carried out based on these conclusions.

The research problem that the thesis explores is "*How can SaaS companies execute small-scale agile development projects in order to maximize customer satisfaction?*". The research problem was further broken down into two research questions in order to facilitate the research process.

Three key conclusions can be drawn from the results of this thesis. These are presented in the upcoming paragraphs and are highlighted in bold.

User stories should be encouraged to represent customer requirements and establish communication with the customer. In order to avoid developing misunderstood customer requirements this thesis suggests that it is important to encourage direct communication between the developer and the customer. User stories can be utilized to document customer requirements and make sure that the project team has a common understanding of the requirements. The customer should be involved in the creation of user stories and should be responsible for defining the priority as well. *Ramesh et. al., (2010),* pointed out that having a shared understanding of the requirements results in less number of defects and post-development changes requested by the customer. Secondly, creating a standard format on which the project team can document customer requirements would ensure uniformity throughout the case company. *Lucassen et. al. (2016)* highlighted that having a common

format for requirements documentation ensures improves the overall productivity of the team and the quality of the user stories.

Pair programming should be encouraged in order to facilitate junior developers in gaining product and customer knowledge. The code base of the product offered by the case company is becoming complex and huge with time. A phenomenon that was also highlighted by Cao and Ramesh (2008) in their research. It can get difficult to understand code behavior when the code base becomes huge and this can also decrease the overall productivity of the team as well *(Hanssen et. al, 2009)*. In order to deepen the knowledge of junior technical consultants the coase company should encourage them to work in pairs with seniors members. This would help them in gaining product-specific knowledge and customer-specific knowledge. Furthermore, working in pairs would make it easier for junior developers to understand the process of small-scale agile development at the case company better and also get a better understanding of good practices that are encouraged at the company. Lastly, working in pairs would also improve the overall quality of the work done for the customer resulting in customer satisfaction.

Guidelines for peer-reviews should be laid-out in order to facilitate developers in maintaining quality of small-scale projects. Based on the results of this thesis it is evident that there is a lack of guidelines regarding the development process that can be used for small-scale projects. Creating guidelines for the development process would ensure that everyone in the case company has a common understanding of the process. It would also make it easier for new developers to understand the process. Firstly, even though the case company has recently started using JIRA there is still a lack of common understanding of how JIRA can be used in small-scale agile development projects. By creating guidelines on how a project team should utilize JIRA in small-scale projects will significantly impact the overall quality of the whole process. JIRA can be used to effectively communicate on important aspects of the project such as deadline, priority, work estimation and requirement documentation. Lastly, there should be guidelines to facilitate the peer-review process. This will help ensure that all projects are peer reviewed before deploying them to customer

environments and will keep a check on the quality of small-scale projects. However, the thesis does not support creating concrete guidelines that would dictate the steps of technical consultants because that can lead to frustration. The suggestion is to create a basic set of guidelines to facilitate the technical consultants in understanding the good practices that should be used when working on small-scale agile projects.

Due to time constraints this thesis was unable to implement the suggested improvements in the case company and record the impact they have on customer satisfaction. One key future research stemming from this thesis can be to observe and record the practical impacts of the above three key conclusions on customer satisfaction. Agile development advocates concise documentation and also encourages to use code as a basis to understand the behaviour of the product. During the literature review of this thesis we found out that there is very little literature that talks about good code commenting practices. Hence, another important area of research here can be to study and create code commenting guidelines for developers. Such guidelines can help in ensuring that the code behaviour is documented in a uniform manner throughout the case company or any other SaaS company.

The most important and interesting future research prospect here can be to study the impact of the recent global pandemic on the good practices that this suggests in order to improve customer satisfaction. The global pandemic came right after the empirical research had already been concluded for this thesis. Hence, it was difficult to account for that aspect within the scope of this thesis. However, it would be interesting to study how the operations of the case company or any other SaaS company were impacted due this sudden pandemic.

Within the scope of this thesis we were unable to account for the customer's point of view regarding small-scale agile development projects. Another key future research area stemming from this thesis can be to reach out to the customers of the case company and study their point of view as well. This would help in understanding the pain points of the customer when it comes to the delivery of small-scale agile projects. It would then help in creating a holistic view of how customer satisfaction can be maximized via such projects

for the case company or generally for any company that provides a SaaS solution to its customers.

References

A. Boden, B. Nett, V. Wulf. (2007) 'Coordination practices in distributed software development of small enterprises', *Second IEEE International Conference on Global Software Engineering (ICGSE 2007)*, IEEE(2007), pp. 235–246.

Aleem, S., Batool, R. and Ahmed, F. (2018) 'Design guidelines for SaaS development process', 2018 IEEE 9th Annual Information Technology, *Electronics and Mobile Communication Conference (IEMCON)* IEEE, pp. 825–831.

B. Selic. (2009) 'Agile Documentation, Anyone?', *IEEE Software*, 26(6), pp. 11-12,

Baskerville, R. and A. T. Wood-Harper. (1996) "A Critical Perspective on Action Research as a Method for Information Systems Research," *Journal of Information Technology*, (11) 3, pp. 235-246.

Baskerville, R. L. (1999) 'Investigating Information Systems with Action Research', *Communications of the Association for Information Systems*, 2(19)

Bearden W.O., Teel J.E. (1983) 'Selected Determinants of Consumer Satisfaction and Complaint Reports', *Journal of Marketing Research*, 20(1), pp. 21-28.

Begel, A. (2008) 'Pair Programming : What ' s in it for Me ?', I*nternational Symposium on Empirical Software Engineering and Measurement*, ESEM 2008, pp. 120–128.

Benlian, A. and Hess, T. (2011) 'Opportunities and risks of software-as-a-service : Findings from a survey of IT executives', *Decision Support Systems*, 52(1), pp. 232–246.

Cao, L. C. L., & Ramesh, B. (2008) 'Agile requirements engineering practices: An empirical study.', *IEEE Software*, 25(1), pp. 60–67.

Cohn, M. (2004) : ‘User stories applied: for agile software development.’, *Addison Wesley, Redwood City*

Dicicco-bloom, B. and Crabtree, B. F. (2006) ‘Making sense of qualitative research The qualitative research interview’, *Medical Education*; 40, pp 314–321.

Dorairaj, S., Noble, J. and Malik, P. (2011) ‘Effective Communication in Distributed Agile Software Development Teams’, *School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand*, pp. 102–116.

Drury-grogan, M. L., Conboy, K. and Acton, T. (2017) ‘Examining decision characteristics & challenges for agile software development’, *The Journal of Systems & Software*,131, pp. 248–265.

Eugene W. Anderson. (1998) ‘Customer Satisfaction and Word of mouth’, *Journal of service research*,

Fitriani, W. R., Rahayu, P. and Sensuse, D. I. (2016) ‘Challenges in Agile Software Development : A Systematic Literature Review’, *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 155–164.

Fornell C. (1992) ‘A National Customer Satisfaction Barometer: The Swedish Experience”, *Journal of Marketing*, 56(1), pp. 6-21.

Fowler, M., Highsmith, J., 2001. The agile manifesto. Software Dev. 9, pp. 28–32.

Gao, Bai, Tsai W. T. and Uehara. (2013) ‘SaaS Testing on Clouds – Issues , Challenges , and Needs’, *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. IEEE, pp. 409–415.

Glaser, B. G. and Strauss, A. L. (1967) ‘The discovery of grounded theory.’, Chicago, IL, USA: Aldine

Green, R., Mazzuchi, T. and Sarkani, S. (2010) 'Communication and Quality in Distributed Agile Development : An Empirical Case Study', *Proceedings of the World Academy of Science, Engineering and Technology (WASET)*, 2010, pp. 322–328.

Grönroos C. (2007) 'Service Management and Marketing: Customer Management in Service Competition', *Wiley*, third edition

H. Kashfi. (2017), "Software Engineering Challenges in Cloud Environment: Software Development Lifecycle Perspective,"

Hanssen, G. K. *et al.* (2009) 'Maintenance and Agile Development : Challenges , Opportunities and Future Directions', *2009 IEEE International Conference on Software Maintenance*. IEEE, (1), pp. 487–490.

Heck, P. and Zaidman, A. (2018) 'A systematic literature review on quality criteria for agile requirements specifications', *Software Quality Journal*.

Helena Holmström , Brian Fitzgerald , Pär J. Ågerfalk & Eoin Ó. Conchúir (2006) 'Agile Practices Reduce Distance in Global Software Development', *Information Systems Management*, 23(3), pp 7-18,

Hulkko, H. and Abrahamsson, P. (2005) 'A Multiple Case Study on the Impact of Pair Programming on Product Quality', *ICSE05: 27th International Conference on Software Engineering*, pp. 495–504.

Hult, M. and S. Lennung. (1980) "Towards A Definition of Action Research: A Note and Bibliography," *Journal of Management Studies*, (17), pp. 241-250.

Ibrahim, Y., Qumer, A. and Al-ani, A. (2016) 'Information & Management Empirical studies of geographically distributed agile development communication challenges : A systematic review', *Information & Management*, 53 (2016), pp. 22–37.

Inayat, I. *et al.* (2015) ‘Computers in Human Behavior A systematic literature review on agile requirements engineering practices and challenges’., *Computers in Human Behavior*, 51, pp. 915–929.

J.D. Herbsleb, D. Moitra. (2001) ‘Global software development’. *IEEE Software,* 18, pp. 16–20.

Kauppinen, M. *et al.* (2009) ‘From Feature Development to Customer Value Creation’, *17th IEEE International Requirements Engineering Conference*, pp. 275-280

Krebs, R., Momm, C. and Kounev, S. (2012) ‘Architectural concerns in multi-tenant SaaS applications’, *2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pp. 426-431

L. Layman, L. Williams, D. Damian, H. Bures. (2006) ‘Essential communication practices for Extreme Programming in a global software development team’, *Inf. Softw. Technol*, 48 (2006), pp. 781–794.

Lee, S. and Yong, H. (2013) ‘Agile Software Development Framework in a Small Project Environment’, *J Inf Process Syst*, 9(1), pp. 69–88.

Loukis, E., Janssen, M. and Mintchev, I. (2019) ‘Determinants of software-as-a-service benefits and impact on firm performance’, *Decision Support Systems*, 117 (December 2018), pp. 38–47.

Lucassen, G., B, F. D. and Werf, J. M. E. M. Van Der (2016) ‘The Use and Effectiveness of User Stories in Practice’, *International Working Conference on Requirements Engineering: Foundation for Software Quality REFSQ 2016*, pp. 205–222.

Lucassen, G., Dalpiaz, F., Werf, J. M. and Brinkkemper S. (2015) ‘Forging High-Quality User Stories : Towards a Discipline for Agile Requirements’, *IEEE 23rd International Requirements Engineering Conference (RE)*, pp. 126–135.

Lui, K. M. and Chan, K. C. C. (2006) 'Pair programming productivity : Novice – novice vs expert – expert', *International Journal of Human-Computer Studies*, 64(2006), pp. 915–925

M. Korkala, M. Pikkarainen, K. Conboy. (2006) 'A case study of customer communication in globally distributed software product development', *Proceedings of the 11th International Conference on Product Focused Software*, ACM(2010), pp. 43–46.

Martin, P.Y. and Turner, B. a. (1986) Grounded Theory and Organizational Research. *The Journal of Applied Behavioral Science*, 22(2), pp. 141-156.

Mehta, N. Steinman, D. and Murphy, L. (2016) *CUSTOMER SUCCESS: How Innovative Companies Are Reducing Churn and Growing Recurring Revenue*, New Jersey: John Wiley & Sons, Inc., Hoboken

Muller, M. (2014) 'Curiosity , Creativity , and Surprise as Analytic Tools : Grounded Theory Method Introduction : Why Use Grounded Theory Method ?', *Springer Science and Business Media New York*, pp. 25–48.

Normann R., Ramirez R. (1993) "From Value Chain to Value Constellation", *Harvard Business Review*, 71(4), pp. 65-77.

Oliver, R.L. (2010) Satisfaction: A Behavioral Perspective on the Consumer, 2nd edn, M.E. Sharpe, Armonk, NY.

P.L. Bannerman, E. Hossain, R. Jeffery. (2012) 'Scrum practice mitigation of global software development coordination challenges: a distinctive advantage?', *45th Hawaii International Conference on System Science (HICSS)*, IEEE(2012), pp. 5309–5318.

Pichler, M. and Wahler, W. (2006) 'Agile Requirements Engineering for a Social Insurance for Occupational Risks Organization : A Case Study', *14th IEEE International Requirements Engineering Conference*

R. Sindhgatta, B. Sengupta, S. Datta. (2011) 'Coping with distance: an empirical study of communication on the jazz platform', P*roceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion,* ACM(2011), pp. 155–162.

R. Vidhyalakshmi and V. Kumar. (2014), 'Design comparison of traditional application and SaaS in Computing for Sustainable Global Development', *2014 International Conference* on 2014, pp. 541–544.

Ramesh, B., Baskerville, R., & Cao, L. (2010). 'Agile requirements engineering practices and challenges: An empirical study.' *Information Systems Journal*, 20(5), pp. 449–480.

S. Nerur and V. Balijepally (2007), Theoretical Reflections on Agile Development Methodologies', *Communication of ACM*, 50(3), pp. 79–83.

Seethamraju, R. (2017) 'Adoption of Software as a Service (SaaS) Enterprise Resource Planning (ERP) Systems in Small and Medium Sized Enterprises (SMEs)', *Information Systems Frontiers*, 17(3), 475–492.

Susman, G. and R. Evered. (1978) "An Assessment of The Scientific Merits of Action Research," *Administrative Science Quarterly*, (23) 4, pp. 582- 603.

Wake, B. (2003) 'INVEST in Good Stories, and SMART Tasks'. http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/ (Accessed: 18 February 2015)

Zhang, X. S. and Dorn, B. (2011) 'Agile Practices in a Small-Scale , Time-Intensive Web Development Project', *2011 Eighth International Conference on Information Technology: New Generations*, pp. 11–12.

Appendices

Appendix 1 Case Interview Questions

Interview Questions for Technical Consultants

Background

1. What is your role and responsibility in the case organization?
    a. How long have you been in the case company?
2. Under the scope of this thesis, the minor development project is defined as 'minor development is referred to as any customer request to modify/improve the existing behavior of their environment in order to improve or introduce new capabilities.'
    a. Is this correct? Or would you like to add something to this definition?

Current Situation

3. How are minor development projects currently handled at the case company? Or
    a. What is the current process that is followed for minor development requests made by the customer?
4. Can you briefly explain the various roles that are involved in a minor development project?
5. What is the current role of a Technical Consultant in minor development projects?
6. Is there a set of standard practices that Technical Consultants follow during minor development projects?
7. How minor development is currently measured and what are the targets?
    a. How do you think this should be done?

Customer Satisfaction/Success

8. Under the scope of this project, customer success is defined as follows 'Customer Success focuses on the general well-being of the existing customers and ensuring that the company does not lose its customer base.'
    a. Would you like to add something to this?
9. What are the most critical components of minor development projects that lead to customer success?
    a. What helps minor development projects to support customer success?

Challenges

10. How do you prioritize your tasks?
    a. For a Technical Consultant: How do you prioritize minor development projects from implementation projects?
    b. For someone from the Technical Services Team: How do you prioritize minor development requests coming from different customers at the same time?
11. What are the current bottlenecks that a Technical Consultant faces during minor development projects?
    a. How do you think these bottlenecks can be mitigated?
12. What challenges does Technical Consultant face that hinder customer success?
    a. How do you think these challenges can be mitigated?
13. How do you think the minor development process should look like?
    a. How can the current process be improved?

Conclusion

14. Is there still something that you would like to point out?
15. How could the interview be improved?

Interview Questions for Customer Success Managers

Background

1. What is your role and responsibility in the case organisation?
    a. How long have you been in the case company?
2. Under the scope of this thesis, the minor development project is defined as 'minor development is referred to as any customer request to modify the existing behavior of their environment in order to improve or introduce new capabilities.'
    a. Is this correct? Or would you like to add something to this definition?

Current Situation

3. How are minor development projects currently handled at the case company? Or
    a. What is the current process that is followed for minor development requests made by the customer?
4. Can you briefly explain the various roles that are involved in a minor development project?

5. Can you please explain your role within the scope of a minor development project at the case organization?
6. How is the success of minor development currently measured and what are the targets?
    a. How do you think this should be done?

Customer Satisfaction/Success

7. Under the scope of this project, customer success is defined as follows 'Customer Success focuses on the general well-being of the existing customers and ensuring that the company does not lose its customer base.'
    a. Would you like to add something to this?
8. Can you briefly explain your role in regards to customer success?
9. What are the most critical components of minor development projects that lead to customer success?
    a. What impact does the success of a minor development request have on Customer Success?

Challenges

10. What are the current bottlenecks that minor development projects face?
    a. How do you think these bottlenecks can be mitigated?
11. What challenges hinder customer success?
    a. How do you think these challenges can be mitigated?
12. How do you think the minor development process should look like?
    a. How can the current process be improved?

Conclusion

13. Is there still something that you would like to point out?
14. How could the interview be improved?